
RsFswp

Release 3.0.1.7

Rohde & Schwarz

Apr 15, 2023

CONTENTS:

1	Revision History	3
1.1	RsFswp	3
1.1.1	Version history	3
2	Getting Started	5
2.1	Introduction	5
2.2	Installation	7
2.3	Finding Available Instruments	9
2.4	Initiating Instrument Session	9
2.5	Plain SCPI Communication	12
2.6	Error Checking	15
2.7	Exception Handling	15
2.8	Transferring Files	17
2.9	Writing Binary Data	17
2.10	Transferring Big Data with Progress	18
2.11	Multithreading	19
2.12	Logging	22
3	Enums	27
3.1	AccessType	27
3.2	AdcPreFilterMode	27
3.3	AdemMeasType	27
3.4	AdjustAlignment	27
3.5	AngleUnit	28
3.6	AnnotationMode	28
3.7	AttenuatorMode	28
3.8	AutoManualMode	28
3.9	AutoManualUserMode	28
3.10	AutoMode	29
3.11	AutoOrOff	29
3.12	AverageModeA	29
3.13	AverageModeB	29
3.14	Band	29
3.15	BandB	30
3.16	BbInputSource	30
3.17	BerRateFormat	30
3.18	BitOrdering	30
3.19	BurstMode	30
3.20	CalibrationScope	31
3.21	CalibrationState	31

3.22	ChannelType	31
3.23	CheckResult	32
3.24	Color	32
3.25	ColorSchemeA	32
3.26	CompatibilityMode	32
3.27	ComponentType	32
3.28	ConfigMode	33
3.29	CorrectionMeasType	33
3.30	CorrectionMethod	33
3.31	CorrectionMode	33
3.32	Counter	33
3.33	CouplingTypeA	34
3.34	CouplingTypeB	34
3.35	DataExportMode	34
3.36	DataFormat	34
3.37	DaysOfWeek	34
3.38	DdemGroup	35
3.39	DdemodFilter	35
3.40	DdemResultType	35
3.41	DdemSignalType	35
3.42	Detect	35
3.43	DetectorB	36
3.44	DetectorC	36
3.45	DiagnosticSignal	36
3.46	DiqUnit	36
3.47	DisplayFormat	36
3.48	DisplayPosition	37
3.49	Duration	37
3.50	DutType	37
3.51	EgateType	37
3.52	EnrType	37
3.53	SpectrumRtype	38
3.54	EventOnce	38
3.55	EvmCalc	38
3.56	Factory	38
3.57	FftFilterMode	38
3.58	FftWindowType	39
3.59	FileFormat	39
3.60	FileFormatDdem	39
3.61	FileSeparator	39
3.62	FilterTypeA	39
3.63	FilterTypeB	40
3.64	FilterTypeC	40
3.65	FilterTypeK91	40
3.66	FineSync	40
3.67	FpeaksSortMode	40
3.68	FrameModulation	41
3.69	FrameModulationB	41
3.70	FramesScope	41
3.71	FrequencyCouplingLinkA	41
3.72	FrequencyType	41
3.73	FunctionA	42
3.74	FunctionB	42
3.75	GatedSourceK30	42

3.76	GeneratorIntf	42
3.77	GeneratorIntfType	42
3.78	GeneratorLink	43
3.79	GpibTerminator	43
3.80	HardcopyContent	43
3.81	HardcopyHeader	43
3.82	HardcopyLogo	43
3.83	HardcopyMode	44
3.84	HardcopyPageSize	44
3.85	HeadersK50	44
3.86	HumsFileFormat	44
3.87	IdnFormat	44
3.88	IfGainMode	45
3.89	IfGainModeDdem	45
3.90	InOutDirection	45
3.91	InputConnectorB	45
3.92	InputConnectorC	45
3.93	InputSelect	46
3.94	InputSource	46
3.95	InputSourceB	46
3.96	InstrumentMode	46
3.97	IqBandwidthMode	46
3.98	IqDataFormat	47
3.99	IqDataFormatDdem	47
3.100	IqRangeType	47
3.101	IqResultDataFormat	47
3.102	IqType	47
3.103	LeftRightDirection	48
3.104	LimitState	48
3.105	LoadType	48
3.106	LoType	48
3.107	LowHigh	48
3.108	MarkerFunctionA	49
3.109	MarkerFunctionB	49
3.110	MarkerMode	49
3.111	MarkerReallImagB	49
3.112	MeasurementStep	49
3.113	MeasurementType	50
3.114	MessageType	50
3.115	MixerIdentifier	50
3.116	MpowerDetector	50
3.117	MskFormat	50
3.118	MspurSearchType	51
3.119	MsSyncMode	51
3.120	NoiseFigureLimit	51
3.121	NoiseFigureResult	51
3.122	NoiseFigureResultCustom	51
3.123	OddEven	52
3.124	OffState	52
3.125	OptimizationCriterion	52
3.126	OptionState	52
3.127	OutputType	52
3.128	PadType	53
3.129	PageMarginUnit	53

3.130	PageOrientation	53
3.131	PathToCalibrate	53
3.132	PictureFormat	53
3.133	PmeterFreqLink	54
3.134	PmRpointMode	54
3.135	PowerMeasFunction	54
3.136	PowerMeterUnit	54
3.137	PowerSource	54
3.138	PowerUnitB	55
3.139	PowerUnitC	55
3.140	PowerUnitDdem	55
3.141	PreamplOption	55
3.142	PresetCompatible	56
3.143	Probability	56
3.144	ProbeMode	56
3.145	ProbeSetupMode	56
3.146	PskFormat	56
3.147	PwrLevelMode	57
3.148	PwrMeasUnit	57
3.149	QamFormat	57
3.150	QpskFormat	57
3.151	RangeClass	57
3.152	RangeParam	58
3.153	RefChannel	58
3.154	ReferenceMode	58
3.155	ReferenceSource	58
3.156	ReferenceSourceA	58
3.157	ReferenceSourceB	59
3.158	ReferenceSourceD	59
3.159	Relay	59
3.160	ResultDevReference	59
3.161	ResultTypeB	60
3.162	ResultTypeC	60
3.163	ResultTypeD	60
3.164	ResultTypeStat	60
3.165	RoscillatorFreqMode	60
3.166	RoscillatorRefOut	61
3.167	ScaleYaxisUnit	61
3.168	ScalingMode	61
3.169	SearchArea	61
3.170	SearchRange	61
3.171	SelectAll	62
3.172	SelectionRangeB	62
3.173	SelectionScope	62
3.174	Separator	62
3.175	SequencerMode	62
3.176	ServiceBandwidth	63
3.177	ServiceState	63
3.178	SidebandPos	63
3.179	SignalLevel	63
3.180	SignalType	63
3.181	SingleValue	64
3.182	Size	64
3.183	SlopeType	64

3.184 SnmpVersion	64
3.185 Source	64
3.186 SourceInt	65
3.187 SpacingY	65
3.188 SpanSetting	65
3.189 State	65
3.190 StatisticMode	65
3.191 StatisticType	66
3.192 Stepsize	66
3.193 StoreType	66
3.194 SubBlockGaps	66
3.195 SummaryMode	66
3.196 SweepModeC	67
3.197 SweepOptimize	67
3.198 SweepType	67
3.199 SymbolSelection	67
3.200 Synchronization	67
3.201 SyncMode	68
3.202 SystemStatus	68
3.203 TechnologyStandardA	68
3.204 TechnologyStandardB	68
3.205 TechnologyStandardDdem	69
3.206 Temperature	69
3.207 TimeFormat	69
3.208 TimeLimitUnit	69
3.209 TimeOrder	69
3.210 TouchscreenState	70
3.211 TraceFormat	70
3.212 TraceModeA	70
3.213 TraceModeB	70
3.214 TraceModeC	70
3.215 TraceModeD	71
3.216 TraceModeE	71
3.217 TraceModeF	71
3.218 TraceModeH	71
3.219 TraceNumber	71
3.220 TracePresetType	72
3.221 TraceReference	72
3.222 TraceRefType	72
3.223 TraceSymbols	72
3.224 TraceTypeDdem	72
3.225 TraceTypeK30	73
3.226 TraceTypeK60	73
3.227 TraceTypeList	73
3.228 TraceTypeNumeric	73
3.229 TriggerOutType	73
3.230 TriggerPort	74
3.231 TriggerSeqSource	74
3.232 TriggerSourceB	74
3.233 TriggerSourceD	74
3.234 TriggerSourceK	75
3.235 TriggerSourceL	75
3.236 TriggerSourceListPower	75
3.237 TriggerSourceMpower	75

3.238	TuningRange	75
3.239	UnitMode	76
3.240	UpDownDirection	76
3.241	UserLevel	76
3.242	WindowDirection	76
3.243	WindowDirReplace	76
3.244	WindowName	77
3.245	WindowTypeBase	77
3.246	WindowTypeIq	77
3.247	WindowTypeK30	77
3.248	WindowTypeK40	78
3.249	WindowTypeK50	78
3.250	WindowTypeK60	78
3.251	WindowTypeK7	78
3.252	WindowTypeK70	79
3.253	Xdistribution	79
3.254	XyDirection	79
4	RepCaps	81
4.1	Channel	81
4.2	Colors	81
4.3	Component	81
4.4	CornerFrequency	82
4.5	DeltaMarker	82
4.6	DisplayLine	82
4.7	ExternalGen	82
4.8	ExternalPort	83
4.9	ExternalRosc	83
4.10	FileList	83
4.11	FilterPy	83
4.12	FreqLine	84
4.13	FreqOffset	84
4.14	GapChannel	84
4.15	GateRange	84
4.16	Generator	85
4.17	InputIx	85
4.18	Item	85
4.19	LimitIx	86
4.20	Line	86
4.21	Marker	86
4.22	MarkerDestination	86
4.23	OutputConnector	87
4.24	Port	87
4.25	PowerClass	87
4.26	PowerMeter	87
4.27	Probe	88
4.28	RangePy	88
4.29	RefMeasurement	88
4.30	Source	88
4.31	SPortPair	89
4.32	Status	89
4.33	Step	89
4.34	Store	89
4.35	SubBlock	90

4.36	SubWindow	90
4.37	TouchStone	90
4.38	Trace	90
4.39	TriggerPort	91
4.40	UpperAltChannel	91
4.41	UserRange	91
4.42	Window	91
4.43	ZoomWindow	92
5	Examples	93
6	RsFswp API Structure	95
6.1	Applications	98
6.1.1	IqAnalyzer	99
6.1.1.1	InputPy<InputIx>	99
6.1.1.1.1	Iq	99
6.1.1.1.1.1	Balanced	100
6.1.1.1.1.2	State	100
6.1.1.1.1.3	Fullscale	101
6.1.1.1.1.4	Level	101
6.1.1.1.1.5	Impedance	102
6.1.1.1.1.6	Ptype	102
6.1.1.1.1.7	Osc	103
6.1.1.1.1.8	Balanced	103
6.1.1.1.1.9	State	103
6.1.1.1.1.10	ConState	104
6.1.1.1.1.11	Coupling	104
6.1.1.1.1.12	Fullscale	105
6.1.1.1.1.13	Level	105
6.1.1.1.1.14	Idn	106
6.1.1.1.1.15	Skew	107
6.1.1.1.1.16	Icomponent	107
6.1.1.1.1.17	Inverted	108
6.1.1.1.1.18	Qcomponent	109
6.1.1.1.1.19	Inverted	109
6.1.1.1.1.20	State	110
6.1.1.1.1.21	Tcpip	111
6.1.1.1.1.22	TypePy	111
6.1.1.1.1.23	Vdevice	112
6.1.1.1.1.24	Vfirmware	113
6.1.1.1.1.25	TypePy	113
6.1.1.2	Layout	114
6.1.1.2.1	Add	114
6.1.1.2.1.1	Window	114
6.1.1.2.2	Catalog	115
6.1.1.2.2.1	Window	115
6.1.1.2.3	Identify	116
6.1.1.2.3.1	Window	116
6.1.1.2.4	Replace	116
6.1.1.2.4.1	Window	117
6.1.1.3	Sense	117
6.1.1.3.1	Iq	118
6.1.1.3.1.1	Fft	118
6.1.1.3.1.2	Algorithm	118

6.1.1.3.1.3	Length	119
6.1.1.3.1.4	Window	120
6.1.1.3.1.5	Length	120
6.1.1.3.1.6	Overlap	121
6.1.1.3.1.7	TypePy	121
6.1.1.3.1.8	Impedance	122
6.1.1.3.1.9	Wband	122
6.1.1.3.2	SwapIq	123
6.1.1.4	Trace	124
6.1.1.4.1	Iq	124
6.1.1.4.1.1	Apcon	124
6.1.1.4.1.2	A	124
6.1.1.4.1.3	B	125
6.1.1.4.1.4	Result	126
6.1.1.4.1.5	State	126
6.1.1.4.1.6	Average	127
6.1.1.4.1.7	Count	127
6.1.1.4.1.8	State	127
6.1.1.4.1.9	Data	128
6.1.1.4.1.10	FormatPy	128
6.1.1.4.1.11	DiqFilter	129
6.1.1.4.1.12	Egate	130
6.1.1.4.1.13	Gap	130
6.1.1.4.1.14	Length	131
6.1.1.4.1.15	Nof	131
6.1.1.4.1.16	State	132
6.1.1.4.1.17	TypePy	132
6.1.1.4.1.18	Eval	133
6.1.1.4.1.19	File	134
6.1.1.4.1.20	Repetition	134
6.1.1.4.1.21	Count	134
6.1.1.4.1.22	Rlength	135
6.1.1.4.1.23	State	136
6.1.1.4.1.24	SymbolRate	137
6.1.1.4.1.25	TpiSample	137
6.1.1.4.1.26	Wband	138
6.1.1.4.1.27	Mbwidth	138
6.1.1.4.1.28	State	139
6.1.1.4.1.29	Wfilter	139
6.1.2	K30_NoiseFigure	140
6.1.2.1	Calculate<Window>	140
6.1.2.1.1	DeltaMarker<DeltaMarker>	141
6.1.2.1.1.1	Aoff	141
6.1.2.1.1.2	Maximum	142
6.1.2.1.1.3	Next	142
6.1.2.1.1.4	Peak	143
6.1.2.1.1.5	Minimum	143
6.1.2.1.1.6	Next	143
6.1.2.1.1.7	Peak	144
6.1.2.1.1.8	Mreference	145
6.1.2.1.1.9	State	146
6.1.2.1.1.10	Trace	147
6.1.2.1.1.11	X	148
6.1.2.1.1.12	Y	149

6.1.2.1.2	Limit<LimitIx>	149
6.1.2.1.2.1	Active	150
6.1.2.1.2.2	Clear	151
6.1.2.1.2.3	Immediate	151
6.1.2.1.2.4	Comment	152
6.1.2.1.2.5	Control	153
6.1.2.1.2.6	Data	153
6.1.2.1.2.7	Shift	154
6.1.2.1.2.8	Copy	154
6.1.2.1.2.9	Fail	155
6.1.2.1.2.10	Lower	156
6.1.2.1.2.11	Data	156
6.1.2.1.2.12	Shift	157
6.1.2.1.2.13	State	158
6.1.2.1.2.14	Name	159
6.1.2.1.2.15	State	160
6.1.2.1.2.16	Trace	161
6.1.2.1.2.17	TypePy	162
6.1.2.1.2.18	Upper	163
6.1.2.1.2.19	Data	163
6.1.2.1.2.20	Shift	164
6.1.2.1.2.21	State	164
6.1.2.1.3	Marker<Marker>	165
6.1.2.1.3.1	Aoff	166
6.1.2.1.3.2	Maximum	166
6.1.2.1.3.3	Next	167
6.1.2.1.3.4	Peak	167
6.1.2.1.3.5	Minimum	168
6.1.2.1.3.6	Next	168
6.1.2.1.3.7	Peak	169
6.1.2.1.3.8	State	169
6.1.2.1.3.9	Trace	170
6.1.2.1.3.10	X	171
6.1.2.1.3.11	Y	172
6.1.2.1.4	Uncertainty	173
6.1.2.1.4.1	Common	173
6.1.2.1.4.2	Data	174
6.1.2.1.4.3	Frequency	174
6.1.2.1.4.4	Gain	175
6.1.2.1.4.5	Noise	176
6.1.2.1.4.6	Results	176
6.1.2.1.4.7	Enr	177
6.1.2.1.4.8	Calibration	178
6.1.2.1.4.9	Uncertainty	178
6.1.2.1.4.10	Cold	179
6.1.2.1.4.11	Hot	180
6.1.2.1.4.12	Uncertainty	180
6.1.2.1.4.13	Cold	181
6.1.2.1.4.14	Hot	182
6.1.2.1.4.15	Match	183
6.1.2.1.4.16	Dut	183
6.1.2.1.4.17	InputPy	184
6.1.2.1.4.18	Rl	184
6.1.2.1.4.19	Vswr	185

6.1.2.1.4.20	Out	185
6.1.2.1.4.21	RI	186
6.1.2.1.4.22	Vswr	186
6.1.2.1.4.23	Preamp	187
6.1.2.1.4.24	RI	187
6.1.2.1.4.25	Vswr	188
6.1.2.1.4.26	Source	189
6.1.2.1.4.27	Calibration	189
6.1.2.1.4.28	RI	189
6.1.2.1.4.29	Sns	190
6.1.2.1.4.30	Vswr	191
6.1.2.1.4.31	RI	192
6.1.2.1.4.32	Sns	192
6.1.2.1.4.33	Vswr	193
6.1.2.1.4.34	Preamp	194
6.1.2.1.4.35	Gain	194
6.1.2.1.4.36	Noise	195
6.1.2.1.4.37	State	196
6.1.2.1.4.38	Result	196
6.1.2.1.4.39	Sanalyzer	197
6.1.2.1.4.40	Gain	197
6.1.2.1.4.41	Uncertainty	197
6.1.2.1.4.42	Noise	198
6.1.2.1.4.43	Uncertainty	198
6.1.2.2	Display	199
6.1.2.2.1	Window<Window>	199
6.1.2.2.1.1	Minfo	199
6.1.2.2.1.2	State	200
6.1.2.2.1.3	Mtable	200
6.1.2.2.1.4	Size	201
6.1.2.2.1.5	Table	202
6.1.2.2.1.6	Item	202
6.1.2.2.1.7	Trace<Trace>	203
6.1.2.2.1.8	Mode	204
6.1.2.2.1.9	Preset	205
6.1.2.2.1.10	Smoothing	206
6.1.2.2.1.11	Aperture	206
6.1.2.2.1.12	State	207
6.1.2.2.1.13	State	208
6.1.2.2.1.14	Symbols	209
6.1.2.2.1.15	Uncertainty	210
6.1.2.2.1.16	X	211
6.1.2.2.1.17	Scale	211
6.1.2.2.1.18	Y	212
6.1.2.2.1.19	Scale	212
6.1.2.2.1.20	Auto	212
6.1.2.2.1.21	Bottom	213
6.1.2.2.1.22	RefLevel	214
6.1.2.2.1.23	Auto	215
6.1.2.2.1.24	Top	216
6.1.2.3	FormatPy	217
6.1.2.3.1	Dexport	217
6.1.2.3.1.1	Cseparator	217
6.1.2.3.1.2	Dseparator	218

6.1.2.3.1.3	FormatPy	219
6.1.2.3.1.4	Header	219
6.1.2.3.1.5	Traces	220
6.1.2.4	Initiate	221
6.1.2.4.1	Continuous	221
6.1.2.4.2	Immediate	222
6.1.2.5	InputPy<InputIx>	222
6.1.2.5.1	Attenuation	223
6.1.2.5.2	Connector	224
6.1.2.5.3	Coupling	224
6.1.2.5.4	Dpath	225
6.1.2.5.5	FilterPy	226
6.1.2.5.5.1	Hpass	226
6.1.2.5.5.2	State	226
6.1.2.5.5.3	Yig	227
6.1.2.5.5.4	State	227
6.1.2.5.6	Gain	228
6.1.2.5.6.1	State	228
6.1.2.5.6.2	Value	229
6.1.2.5.7	Impedance	230
6.1.2.5.8	TypePy	231
6.1.2.6	Layout	232
6.1.2.6.1	Add	232
6.1.2.6.1.1	Window	232
6.1.2.6.2	Catalog	233
6.1.2.6.2.1	Window	233
6.1.2.6.3	Identify	234
6.1.2.6.3.1	Window	234
6.1.2.6.4	Move	234
6.1.2.6.4.1	Window	235
6.1.2.6.5	Remove	235
6.1.2.6.5.1	Window	236
6.1.2.6.6	Replace	236
6.1.2.6.6.1	Window	236
6.1.2.6.7	Splitter	237
6.1.2.7	MassMemory	238
6.1.2.7.1	Store<Store>	238
6.1.2.7.1.1	Trace	239
6.1.2.8	Output<OutputConnector>	239
6.1.2.8.1	Trigger<TriggerPort>	240
6.1.2.8.1.1	Direction	240
6.1.2.8.1.2	Level	241
6.1.2.8.1.3	Otype	242
6.1.2.8.1.4	Pulse	243
6.1.2.8.1.5	Immediate	243
6.1.2.8.1.6	Length	244
6.1.2.9	Sense	245
6.1.2.9.1	Bandwidth	245
6.1.2.9.1.1	ListPy	245
6.1.2.9.1.2	Data	246
6.1.2.9.2	Configure	247
6.1.2.9.2.1	Control	247
6.1.2.9.2.2	Correction	248
6.1.2.9.2.3	Frequency	249

6.1.2.9.2.4	Continuous	249
6.1.2.9.2.5	Single	250
6.1.2.9.2.6	ListPy	250
6.1.2.9.2.7	Continuous	250
6.1.2.9.2.8	Single	251
6.1.2.9.2.9	Measurement	252
6.1.2.9.2.10	Mode	252
6.1.2.9.2.11	Dut	253
6.1.2.9.2.12	System	253
6.1.2.9.2.13	Ifreq	254
6.1.2.9.2.14	Frequency	254
6.1.2.9.2.15	Lo	255
6.1.2.9.2.16	Frequency	255
6.1.2.9.2.17	Single	256
6.1.2.9.3	Correction	257
6.1.2.9.3.1	Enr	257
6.1.2.9.3.2	Calibration	257
6.1.2.9.3.3	Mode	258
6.1.2.9.3.4	Sns	259
6.1.2.9.3.5	SrNumber	259
6.1.2.9.3.6	Spot	260
6.1.2.9.3.7	Cold	260
6.1.2.9.3.8	Hot	261
6.1.2.9.3.9	Table	262
6.1.2.9.3.10	Select	262
6.1.2.9.3.11	TypePy	263
6.1.2.9.3.12	Common	264
6.1.2.9.3.13	Measurement	264
6.1.2.9.3.14	Mode	265
6.1.2.9.3.15	Sns	265
6.1.2.9.3.16	SrNumber	266
6.1.2.9.3.17	Spot	266
6.1.2.9.3.18	Cold	267
6.1.2.9.3.19	Hot	268
6.1.2.9.3.20	Table	268
6.1.2.9.3.21	Data	269
6.1.2.9.3.22	ListPy	270
6.1.2.9.3.23	Select	270
6.1.2.9.3.24	Temperature	271
6.1.2.9.3.25	Data	271
6.1.2.9.3.26	ListPy	272
6.1.2.9.3.27	TypePy	273
6.1.2.9.3.28	Sns	273
6.1.2.9.3.29	Auto	274
6.1.2.9.3.30	SrNumber	274
6.1.2.9.3.31	State	274
6.1.2.9.3.32	Irejection	275
6.1.2.9.3.33	Loss	275
6.1.2.9.3.34	Calibration	276
6.1.2.9.3.35	Mode	276
6.1.2.9.3.36	Spot	277
6.1.2.9.3.37	Table	277
6.1.2.9.3.38	Data	278
6.1.2.9.3.39	ListPy	279

6.1.2.9.3.40	Select	279
6.1.2.9.3.41	Temperature	280
6.1.2.9.3.42	InputPy	280
6.1.2.9.3.43	Mode	280
6.1.2.9.3.44	Spot	281
6.1.2.9.3.45	Table	282
6.1.2.9.3.46	Data	282
6.1.2.9.3.47	ListPy	283
6.1.2.9.3.48	Select	283
6.1.2.9.3.49	Temperature	284
6.1.2.9.3.50	Output	285
6.1.2.9.3.51	Mode	285
6.1.2.9.3.52	Spot	286
6.1.2.9.3.53	Table	286
6.1.2.9.3.54	Data	287
6.1.2.9.3.55	ListPy	288
6.1.2.9.3.56	Select	288
6.1.2.9.3.57	Temperature	289
6.1.2.9.3.58	Save	289
6.1.2.9.3.59	State	290
6.1.2.9.3.60	Temperature	290
6.1.2.9.3.61	Control	291
6.1.2.9.4	Frequency	292
6.1.2.9.4.1	Center	292
6.1.2.9.4.2	Points	292
6.1.2.9.4.3	Single	293
6.1.2.9.4.4	Coupled	294
6.1.2.9.4.5	Span	294
6.1.2.9.4.6	Start	295
6.1.2.9.4.7	Step	296
6.1.2.9.4.8	Stop	296
6.1.2.9.4.9	Table	297
6.1.2.9.4.10	Data	297
6.1.2.9.5	Probe<Probe>	298
6.1.2.9.5.1	Id	298
6.1.2.9.5.2	PartNumber	298
6.1.2.9.5.3	SrNumber	299
6.1.2.9.6	Sweep	299
6.1.2.9.6.1	Count	300
6.1.2.9.6.2	Egate	300
6.1.2.9.6.3	Auto	301
6.1.2.9.6.4	Continuous	302
6.1.2.9.6.5	Pcount	302
6.1.2.9.6.6	Plength	303
6.1.2.9.6.7	State	303
6.1.2.9.6.8	Holdoff	304
6.1.2.9.6.9	Length	304
6.1.2.9.6.10	Level	305
6.1.2.9.6.11	External<ExternalPort>	305
6.1.2.9.6.12	Polarity	306
6.1.2.9.6.13	Source	307
6.1.2.9.6.14	TypePy	308
6.1.2.9.6.15	Time	308
6.1.2.9.6.16	Auto	309

6.1.2.10	Source	309
6.1.2.10.1	Current	310
6.1.2.10.1.1	Auxiliary	310
6.1.2.10.1.2	Limit	310
6.1.2.10.1.3	High	311
6.1.2.10.1.4	Control<Source>	311
6.1.2.10.1.5	Limit	311
6.1.2.10.1.6	High	312
6.1.2.10.1.7	Power<Source>	312
6.1.2.10.1.8	Limit	313
6.1.2.10.1.9	High	313
6.1.2.10.1.10	Sequence	314
6.1.2.10.1.11	Result	314
6.1.2.10.2	External	314
6.1.2.10.2.1	Frequency	315
6.1.2.10.2.2	Factor	315
6.1.2.10.2.3	Denominator	315
6.1.2.10.2.4	Numerator	316
6.1.2.10.2.5	Offset<FreqOffset>	317
6.1.2.10.2.6	Power	318
6.1.2.10.2.7	Level	318
6.1.2.10.2.8	Roscillator	319
6.1.2.10.2.9	Source	319
6.1.2.10.3	Generator	320
6.1.2.10.3.1	Channel	320
6.1.2.10.3.2	Coupling	320
6.1.2.10.3.3	DutBypass	321
6.1.2.10.3.4	Frequency	321
6.1.2.10.3.5	Level	322
6.1.2.10.3.6	Modulation	322
6.1.2.10.3.7	Pulse	323
6.1.2.10.3.8	Period	323
6.1.2.10.3.9	Trigger	324
6.1.2.10.3.10	Output	324
6.1.2.10.3.11	Width	325
6.1.2.10.3.12	State	325
6.1.2.10.4	Nsource	326
6.1.2.10.4.1	State	326
6.1.2.10.5	Power	327
6.1.2.10.5.1	Sequence	327
6.1.2.10.5.2	Result	327
6.1.2.10.6	Voltage	328
6.1.2.10.6.1	Auxiliary	328
6.1.2.10.6.2	Level	328
6.1.2.10.6.3	Amplitude	328
6.1.2.10.6.4	Limit	329
6.1.2.10.6.5	High	329
6.1.2.10.6.6	Low	330
6.1.2.10.6.7	State	330
6.1.2.10.6.8	Channel	331
6.1.2.10.6.9	Coupling	331
6.1.2.10.6.10	Control<Source>	332
6.1.2.10.6.11	Level	332
6.1.2.10.6.12	Amplitude	333

6.1.2.10.6.13	Limit	333
6.1.2.10.6.14	High	334
6.1.2.10.6.15	Low	335
6.1.2.10.6.16	State	335
6.1.2.10.6.17	Power<Source>	336
6.1.2.10.6.18	Level	336
6.1.2.10.6.19	Amplitude	337
6.1.2.10.6.20	Limit	338
6.1.2.10.6.21	High	338
6.1.2.10.6.22	Low	339
6.1.2.10.6.23	Mode	339
6.1.2.10.6.24	State	340
6.1.2.10.6.25	Limit	341
6.1.2.10.6.26	High	341
6.1.2.10.6.27	Sequence	342
6.1.2.10.6.28	Result	342
6.1.2.10.6.29	State	342
6.1.2.10.6.30	UsePort	343
6.1.2.11	System	344
6.1.2.11.1	Communicate	344
6.1.2.11.1.1	Gpib	344
6.1.2.11.1.2	Rdevice	344
6.1.2.11.1.3	Generator	345
6.1.2.11.1.4	Address	345
6.1.2.11.1.5	Rdevice	346
6.1.2.11.1.6	Generator	346
6.1.2.11.1.7	Interface	346
6.1.2.11.1.8	Link	347
6.1.2.11.1.9	TypePy	347
6.1.2.11.1.10	Tcpip	348
6.1.2.11.1.11	Rdevice	348
6.1.2.11.1.12	Generator	349
6.1.2.11.1.13	Address	349
6.1.2.11.2	Configure	350
6.1.2.11.2.1	Dut	350
6.1.2.11.2.2	Gain	350
6.1.2.11.2.3	Stime	351
6.1.2.11.2.4	Generator	351
6.1.2.11.2.5	Control	352
6.1.2.11.2.6	State	352
6.1.2.11.2.7	Initialise	353
6.1.2.11.2.8	Auto	353
6.1.2.11.2.9	Immediate	354
6.1.2.11.2.10	Switch	354
6.1.2.11.2.11	Auto	355
6.1.2.12	Trace<Window>	355
6.1.2.12.1	Data	356
6.1.2.13	Trigger<TriggerPort>	357
6.1.2.13.1	Sequence	357
6.1.2.13.1.1	Dtime	357
6.1.2.13.1.2	Holdoff	358
6.1.2.13.1.3	Time	358
6.1.2.13.1.4	Level	359
6.1.2.13.1.5	External<ExternalPort>	359

	6.1.2.13.1.6	Slope	361
	6.1.2.13.1.7	Source	362
6.1.3	K40_PhaseNoise		363
6.1.3.1	Calculate<Window>		363
	6.1.3.1.1	DeltaMarker<DeltaMarker>	363
	6.1.3.1.1.1	Y	364
	6.1.3.1.2	Limit<LimitIx>	364
	6.1.3.1.2.1	Active	365
	6.1.3.1.3	Marker<Marker>	365
	6.1.3.1.3.1	Y	366
	6.1.3.1.4	PnLimit	366
	6.1.3.1.4.1	Auto	367
	6.1.3.1.4.2	Fail	367
	6.1.3.1.4.3	Fc<CornerFrequency>	368
	6.1.3.1.5	Snoise<Marker>	369
	6.1.3.1.5.1	Decades	369
	6.1.3.1.5.2	X	369
	6.1.3.1.5.3	Y	370
	6.1.3.1.5.4	Y	371
6.1.3.2	Display		371
	6.1.3.2.1	Window<Window>	371
	6.1.3.2.1.1	Trace<Trace>	372
	6.1.3.2.1.2	Select	372
6.1.3.3	Fetch		373
	6.1.3.3.1	Pnoise<Trace>	373
	6.1.3.3.1.1	Ipn	373
	6.1.3.3.1.2	Measured	374
	6.1.3.3.1.3	Frequency	374
	6.1.3.3.1.4	Level	375
	6.1.3.3.1.5	Rfm	375
	6.1.3.3.1.6	Rms	376
	6.1.3.3.1.7	Rpm	376
	6.1.3.3.1.8	Spurs	377
	6.1.3.3.1.9	Discrete	377
	6.1.3.3.1.10	Random	378
	6.1.3.3.1.11	Sweep	378
	6.1.3.3.1.12	Avg	378
	6.1.3.3.1.13	Fdrift	379
	6.1.3.3.1.14	Ldrift	379
	6.1.3.3.1.15	Mdrift	380
	6.1.3.3.1.16	Start	380
	6.1.3.3.1.17	Stop	381
	6.1.3.3.1.18	SymbolRate	381
	6.1.3.3.1.19	User<UserRange>	382
	6.1.3.3.1.20	Ipn	382
	6.1.3.3.1.21	Rfm	383
	6.1.3.3.1.22	Rms	383
	6.1.3.3.1.23	Rpm	384
6.1.3.4	Layout		384
	6.1.3.4.1	Add	384
	6.1.3.4.1.1	Window	385
	6.1.3.4.2	Catalog	386
	6.1.3.4.2.1	Window	386
	6.1.3.4.3	Identify	386

6.1.3.4.3.1	Window	387
6.1.3.4.4	Replace	387
6.1.3.4.4.1	Window	387
6.1.3.5	Sense	388
6.1.3.5.1	Probe<Probe>	388
6.1.3.5.1.1	Id	389
6.1.3.5.1.2	SrNumber	389
6.1.3.6	Trace<Window>	390
6.1.3.6.1	Data	390
6.1.4	K50_Spurious	391
6.1.4.1	Calculate<Window>	391
6.1.4.1.1	DeltaMarker<DeltaMarker>	392
6.1.4.1.1.1	Aoff	392
6.1.4.1.1.2	LinkTo	393
6.1.4.1.1.3	Marker<MarkerDestination>	393
6.1.4.1.1.4	Maximum	395
6.1.4.1.1.5	Left	395
6.1.4.1.1.6	Next	396
6.1.4.1.1.7	Peak	396
6.1.4.1.1.8	Right	397
6.1.4.1.1.9	Minimum	397
6.1.4.1.1.10	Left	397
6.1.4.1.1.11	Next	398
6.1.4.1.1.12	Peak	399
6.1.4.1.1.13	Right	399
6.1.4.1.1.14	Mref	400
6.1.4.1.1.15	State	401
6.1.4.1.1.16	Trace	402
6.1.4.1.1.17	X	403
6.1.4.1.1.18	Relative	404
6.1.4.1.1.19	Y	404
6.1.4.1.2	Dline<DisplayLine>	405
6.1.4.1.2.1	State	406
6.1.4.1.3	Fline<FreqLine>	407
6.1.4.1.3.1	State	408
6.1.4.1.4	Limit<LimitIx>	409
6.1.4.1.4.1	Clear	409
6.1.4.1.4.2	Immediate	410
6.1.4.1.4.3	Fail	410
6.1.4.1.5	Marker<Marker>	411
6.1.4.1.5.1	Aoff	411
6.1.4.1.5.2	Link	412
6.1.4.1.5.3	LinkTo	412
6.1.4.1.5.4	Marker<MarkerDestination>	413
6.1.4.1.5.5	Maximum	414
6.1.4.1.5.6	Left	414
6.1.4.1.5.7	Next	415
6.1.4.1.5.8	Peak	416
6.1.4.1.5.9	Right	416
6.1.4.1.5.10	Minimum	417
6.1.4.1.5.11	Left	417
6.1.4.1.5.12	Next	418
6.1.4.1.5.13	Peak	418
6.1.4.1.5.14	Right	419

6.1.4.1.5.15	Pexcursion	419
6.1.4.1.5.16	State	420
6.1.4.1.5.17	Trace	421
6.1.4.1.5.18	X	422
6.1.4.1.5.19	Y	423
6.1.4.1.6	PeakSearch	423
6.1.4.1.6.1	Pshow	423
6.1.4.1.7	Pmeter<PowerMeter>	424
6.1.4.1.7.1	Relative	425
6.1.4.1.7.2	Magnitude	425
6.1.4.1.7.3	Auto	426
6.1.4.1.7.4	State	427
6.1.4.1.8	Ssearch	428
6.1.4.1.8.1	Table	428
6.1.4.1.8.2	Column	428
6.1.4.2	Calibration	429
6.1.4.2.1	Pmeter<PowerMeter>	429
6.1.4.2.1.1	Zero	430
6.1.4.2.1.2	Auto	430
6.1.4.3	Display	431
6.1.4.3.1	Mtable	431
6.1.4.3.2	Window<Window>	431
6.1.4.3.2.1	Minfo	432
6.1.4.3.2.2	State	432
6.1.4.3.2.3	Mtable	433
6.1.4.3.2.4	Pmeter<PowerMeter>	434
6.1.4.3.2.5	State	434
6.1.4.3.2.6	Size	435
6.1.4.3.2.7	Trace<Trace>	436
6.1.4.3.2.8	Length	436
6.1.4.3.2.9	State	437
6.1.4.3.2.10	Y	438
6.1.4.3.2.11	Scale	438
6.1.4.3.2.12	Auto	439
6.1.4.3.2.13	Maximum	440
6.1.4.3.2.14	Minimum	440
6.1.4.3.2.15	Pdivision	441
6.1.4.3.2.16	RefPosition	442
6.1.4.3.2.17	Rvalue	442
6.1.4.3.3	Wselect	443
6.1.4.4	Fetch	444
6.1.4.4.1	Pmeter<PowerMeter>	444
6.1.4.5	FormatPy	445
6.1.4.5.1	Dexport	445
6.1.4.5.1.1	Dseparator	445
6.1.4.5.1.2	Header	446
6.1.4.5.1.3	Traces	447
6.1.4.6	Initiate	447
6.1.4.6.1	ConMeas	448
6.1.4.6.2	Continuous	448
6.1.4.6.3	Immediate	449
6.1.4.6.4	Spurious	450
6.1.4.6.5	Sync	450
6.1.4.7	InputPy	451

6.1.4.7.1	Connector	451
6.1.4.7.2	Coupling	452
6.1.4.7.3	FilterPy	452
6.1.4.7.3.1	Hpass	452
6.1.4.7.3.2	State	453
6.1.4.7.3.3	Yig	454
6.1.4.7.3.4	State	454
6.1.4.7.4	Impedance	454
6.1.4.8	Layout	455
6.1.4.8.1	Add	455
6.1.4.8.1.1	Window	456
6.1.4.8.2	Catalog	456
6.1.4.8.2.1	Window	457
6.1.4.8.3	Identify	457
6.1.4.8.3.1	Window	457
6.1.4.8.4	Move	458
6.1.4.8.4.1	Window	458
6.1.4.8.5	Remove	459
6.1.4.8.5.1	Window	459
6.1.4.8.6	Replace	459
6.1.4.8.6.1	Window	460
6.1.4.8.7	Splitter	460
6.1.4.9	MassMemory	461
6.1.4.9.1	Store<Store>	461
6.1.4.9.1.1	Spur	462
6.1.4.9.1.2	Meas	462
6.1.4.9.1.3	Table	462
6.1.4.9.1.4	Trace	463
6.1.4.10	Output	464
6.1.4.10.1	Trigger<TriggerPort>	464
6.1.4.10.1.1	Direction	464
6.1.4.10.1.2	Level	465
6.1.4.10.1.3	Otype	466
6.1.4.10.1.4	Pulse	467
6.1.4.10.1.5	Immediate	467
6.1.4.10.1.6	Length	467
6.1.4.11	Read	468
6.1.4.11.1	Pmeter<PowerMeter>	468
6.1.4.12	Sense	469
6.1.4.12.1	Adjust	469
6.1.4.12.1.1	Carrier	470
6.1.4.12.1.2	Level	470
6.1.4.12.2	Correction	471
6.1.4.12.2.1	Cvl	471
6.1.4.12.2.2	Band	472
6.1.4.12.2.3	Bias	473
6.1.4.12.2.4	Catalog	473
6.1.4.12.2.5	Comment	474
6.1.4.12.2.6	Data	475
6.1.4.12.2.7	Harmonic	475
6.1.4.12.2.8	Mixer	476
6.1.4.12.2.9	Ports	477
6.1.4.12.2.10	Select	478
6.1.4.12.2.11	Snumber	478

6.1.4.12.3	Creference	479
6.1.4.12.3.1	Freference	479
6.1.4.12.3.2	Frequency	480
6.1.4.12.3.3	Guard	481
6.1.4.12.3.4	Interval	481
6.1.4.12.3.5	State	482
6.1.4.12.3.6	Harmonics	482
6.1.4.12.3.7	Identify	482
6.1.4.12.3.8	Mnumber	483
6.1.4.12.3.9	Tolerance	484
6.1.4.12.3.10	Pdetect	484
6.1.4.12.3.11	Range	484
6.1.4.12.3.12	Center	485
6.1.4.12.3.13	Span	485
6.1.4.12.3.14	Start	486
6.1.4.12.3.15	Stop	487
6.1.4.12.3.16	Preference	487
6.1.4.12.3.17	Srange	488
6.1.4.12.3.18	Value	488
6.1.4.12.4	Directed	489
6.1.4.12.4.1	InputPy	490
6.1.4.12.4.2	Attenuation	490
6.1.4.12.4.3	Gain	491
6.1.4.12.4.4	State	491
6.1.4.12.4.5	Value	492
6.1.4.12.4.6	Loffset	492
6.1.4.12.4.7	MfRbw	493
6.1.4.12.4.8	Nfft	493
6.1.4.12.4.9	Pexcursion	494
6.1.4.12.4.10	RefLevel	494
6.1.4.12.4.11	Settings	495
6.1.4.12.5	Fplan	496
6.1.4.12.5.1	Component<Component>	497
6.1.4.12.5.2	Add	498
6.1.4.12.5.3	Bcenter	498
6.1.4.12.5.4	Bspan	499
6.1.4.12.5.5	Factor	500
6.1.4.12.5.6	Identity	500
6.1.4.12.5.7	Port<Port>	501
6.1.4.12.5.8	Frequency	502
6.1.4.12.5.9	Mharmonic	503
6.1.4.12.5.10	TypePy	504
6.1.4.12.5.11	Predicted	505
6.1.4.12.5.12	Transfer	505
6.1.4.12.6	Frequency	506
6.1.4.12.6.1	Offset	506
6.1.4.12.7	ListPy	506
6.1.4.12.7.1	Range<RangePy>	508
6.1.4.12.7.2	Bandwidth	508
6.1.4.12.7.3	Auto	509
6.1.4.12.7.4	Resolution	510
6.1.4.12.7.5	Count	510
6.1.4.12.7.6	Frequency	511
6.1.4.12.7.7	Start	511

6.1.4.12.7.8	Stop	512
6.1.4.12.7.9	InputPy	513
6.1.4.12.7.10	Attenuation	513
6.1.4.12.7.11	Gain	514
6.1.4.12.7.12	State	514
6.1.4.12.7.13	Value	515
6.1.4.12.7.14	Insert	516
6.1.4.12.7.15	Loffset	516
6.1.4.12.7.16	MfRbw	517
6.1.4.12.7.17	Nfft	518
6.1.4.12.7.18	Pexcursion	518
6.1.4.12.7.19	RefLevel	519
6.1.4.12.7.20	SnRatio	520
6.1.4.12.7.21	Threshold	521
6.1.4.12.7.22	Start	521
6.1.4.12.7.23	Stop	522
6.1.4.12.7.24	UaRange	522
6.1.4.12.8	Measure	523
6.1.4.12.8.1	Points	523
6.1.4.12.9	Mixer	524
6.1.4.12.9.1	Bias	524
6.1.4.12.9.2	High	524
6.1.4.12.9.3	Low	525
6.1.4.12.9.4	Frequency	525
6.1.4.12.9.5	Handover	526
6.1.4.12.9.6	Start	526
6.1.4.12.9.7	Stop	527
6.1.4.12.9.8	Harmonic	527
6.1.4.12.9.9	Band	528
6.1.4.12.9.10	High	529
6.1.4.12.9.11	State	530
6.1.4.12.9.12	Value	530
6.1.4.12.9.13	Low	531
6.1.4.12.9.14	TypePy	531
6.1.4.12.9.15	LoPower	532
6.1.4.12.9.16	Loss	533
6.1.4.12.9.17	High	533
6.1.4.12.9.18	Low	533
6.1.4.12.9.19	Table	534
6.1.4.12.9.20	High	534
6.1.4.12.9.21	Low	535
6.1.4.12.9.22	Ports	536
6.1.4.12.9.23	RfOverrange	536
6.1.4.12.9.24	State	536
6.1.4.12.9.25	Signal	537
6.1.4.12.9.26	State	538
6.1.4.12.9.27	Threshold	538
6.1.4.12.10	Pmeter<PowerMeter>	539
6.1.4.12.10.1	Dcycle	539
6.1.4.12.10.2	State	539
6.1.4.12.10.3	Value	540
6.1.4.12.10.4	Frequency	541
6.1.4.12.10.5	Link	542
6.1.4.12.10.6	Mtime	543

6.1.4.12.10.7	Average	543
6.1.4.12.10.8	Count	544
6.1.4.12.10.9	State	545
6.1.4.12.10.10	Roffset	545
6.1.4.12.10.11	State	546
6.1.4.12.10.12	Soffset	546
6.1.4.12.10.13	State	547
6.1.4.12.10.14	Trigger	548
6.1.4.12.10.15	Level	548
6.1.4.12.10.16	State	549
6.1.4.12.10.17	Update	550
6.1.4.12.10.18	State	550
6.1.4.12.11	Probe<Probe>	551
6.1.4.12.11.1	Id	551
6.1.4.12.11.2	PartNumber	551
6.1.4.12.11.3	SrNumber	552
6.1.4.12.11.4	Setup	552
6.1.4.12.11.5	AttRatio	552
6.1.4.12.11.6	CmOffset	553
6.1.4.12.11.7	DmOffset	554
6.1.4.12.11.8	Mode	555
6.1.4.12.11.9	Name	555
6.1.4.12.11.10	NmOffset	556
6.1.4.12.11.11	Pmode	557
6.1.4.12.11.12	PmOffset	557
6.1.4.12.11.13	State	558
6.1.4.12.11.14	TypePy	559
6.1.4.12.12	Ssearch	559
6.1.4.12.12.1	Control	559
6.1.4.12.12.2	Fplan	560
6.1.4.12.12.3	Tolerance	561
6.1.4.12.12.4	Mspur	561
6.1.4.12.12.5	Rmark	562
6.1.4.12.12.6	Rremove	563
6.1.4.12.12.7	Stype	563
6.1.4.12.13	Transfer	564
6.1.4.12.13.1	Segment	564
6.1.4.12.13.2	Spur	565
6.1.4.13	Trace<Window>	565
6.1.4.13.1	Data	566
6.1.4.13.1.1	X	566
6.1.4.14	Trigger	567
6.1.4.14.1	Sequence	567
6.1.4.14.1.1	Dtime	568
6.1.4.14.1.2	Holdoff	568
6.1.4.14.1.3	Time	568
6.1.4.14.1.4	IfPower	569
6.1.4.14.1.5	Holdoff	569
6.1.4.14.1.6	Hysteresis	570
6.1.4.14.1.7	Level	571
6.1.4.14.1.8	External<ExternalPort>	571
6.1.4.14.1.9	IfPower	572
6.1.4.14.1.10	IqPower	573
6.1.4.14.1.11	RfPower	573

6.1.4.14.1.12	Slope	574
6.1.4.14.1.13	Source	575
6.1.4.15	TriggerInvoke	575
6.1.4.16	Unit	576
6.1.4.16.1	Pmeter<PowerMeter>	576
6.1.4.16.1.1	Power	577
6.1.4.16.1.2	Ratio	578
6.1.5	K60_Transient	578
6.1.5.1	Calculate<Window>	579
6.1.5.1.1	ChrDetection	579
6.1.5.1.1.1	Pnoise	579
6.1.5.1.1.2	Frequency	580
6.1.5.1.1.3	Start	580
6.1.5.1.1.4	Stop	581
6.1.5.1.1.5	Length	582
6.1.5.1.1.6	Offset	582
6.1.5.1.1.7	Begin	583
6.1.5.1.1.8	End	584
6.1.5.1.1.9	Reference	584
6.1.5.1.2	DeltaMarker<DeltaMarker>	585
6.1.5.1.2.1	Aoff	586
6.1.5.1.2.2	LinkTo	586
6.1.5.1.2.3	Marker<MarkerDestination>	587
6.1.5.1.2.4	Maximum	588
6.1.5.1.2.5	Left	588
6.1.5.1.2.6	Next	589
6.1.5.1.2.7	Peak	590
6.1.5.1.2.8	Right	590
6.1.5.1.2.9	Minimum	591
6.1.5.1.2.10	Left	591
6.1.5.1.2.11	Next	592
6.1.5.1.2.12	Peak	592
6.1.5.1.2.13	Right	593
6.1.5.1.2.14	Spectrogram	593
6.1.5.1.2.15	Frame	593
6.1.5.1.2.16	Sarea	595
6.1.5.1.2.17	Xy	596
6.1.5.1.2.18	Maximum	596
6.1.5.1.2.19	Peak	596
6.1.5.1.2.20	Minimum	597
6.1.5.1.2.21	Peak	597
6.1.5.1.2.22	Y	598
6.1.5.1.2.23	Maximum	598
6.1.5.1.2.24	Above	598
6.1.5.1.2.25	Below	599
6.1.5.1.2.26	Next	599
6.1.5.1.2.27	Peak	600
6.1.5.1.2.28	Minimum	601
6.1.5.1.2.29	Above	601
6.1.5.1.2.30	Below	602
6.1.5.1.2.31	Next	602
6.1.5.1.2.32	Peak	603
6.1.5.1.2.33	State	603
6.1.5.1.2.34	Trace	604

6.1.5.1.2.35	X	605
6.1.5.1.2.36	Relative	606
6.1.5.1.2.37	Y	607
6.1.5.1.3	Distribution	607
6.1.5.1.3.1	Nbins	608
6.1.5.1.4	HopDetection	608
6.1.5.1.4.1	Pnoise	609
6.1.5.1.4.2	Frequency	609
6.1.5.1.4.3	Start	609
6.1.5.1.4.4	Stop	610
6.1.5.1.4.5	Length	611
6.1.5.1.4.6	Offset	612
6.1.5.1.4.7	Begin	612
6.1.5.1.4.8	End	613
6.1.5.1.4.9	Reference	613
6.1.5.1.5	Marker<Marker>	614
6.1.5.1.5.1	Aoff	615
6.1.5.1.5.2	LinkTo	615
6.1.5.1.5.3	Marker<MarkerDestination>	616
6.1.5.1.5.4	Maximum	617
6.1.5.1.5.5	Left	618
6.1.5.1.5.6	Next	618
6.1.5.1.5.7	Peak	619
6.1.5.1.5.8	Right	619
6.1.5.1.5.9	Minimum	620
6.1.5.1.5.10	Left	620
6.1.5.1.5.11	Next	621
6.1.5.1.5.12	Peak	621
6.1.5.1.5.13	Right	622
6.1.5.1.5.14	Pexcursion	622
6.1.5.1.5.15	Spectrogram	623
6.1.5.1.5.16	Frame	623
6.1.5.1.5.17	Sarea	624
6.1.5.1.5.18	Xy	625
6.1.5.1.5.19	Maximum	625
6.1.5.1.5.20	Peak	625
6.1.5.1.5.21	Minimum	626
6.1.5.1.5.22	Peak	626
6.1.5.1.5.23	Y	627
6.1.5.1.5.24	Maximum	627
6.1.5.1.5.25	Above	627
6.1.5.1.5.26	Below	628
6.1.5.1.5.27	Next	628
6.1.5.1.5.28	Peak	629
6.1.5.1.5.29	Minimum	629
6.1.5.1.5.30	Above	630
6.1.5.1.5.31	Below	630
6.1.5.1.5.32	Next	631
6.1.5.1.5.33	Peak	631
6.1.5.1.5.34	State	632
6.1.5.1.5.35	Trace	633
6.1.5.1.5.36	X	634
6.1.5.1.5.37	Y	635
6.1.5.1.6	Spectrogram	635

6.1.5.1.6.1	Frame	636
6.1.5.1.6.2	Count	636
6.1.5.1.6.3	Select	637
6.1.5.1.6.4	Hdepth	637
6.1.5.1.6.5	Tstamp	638
6.1.5.1.6.6	Data	638
6.1.5.1.6.7	State	639
6.1.5.2	Display	640
6.1.5.2.1	Mtable	640
6.1.5.2.2	Window<Window>	641
6.1.5.2.2.1	Size	641
6.1.5.2.2.2	Spectrogram	642
6.1.5.2.2.3	Color	642
6.1.5.2.2.4	Style	643
6.1.5.2.2.5	Trace<Trace>	644
6.1.5.2.2.6	Mode	644
6.1.5.2.2.7	Hcontinuous	645
6.1.5.2.2.8	Preset	646
6.1.5.2.2.9	State	647
6.1.5.2.2.10	Y	648
6.1.5.2.2.11	Scale	648
6.1.5.2.2.12	RefLevel	649
6.1.5.2.2.13	Offset	650
6.1.5.2.2.14	RefPosition	651
6.1.5.2.2.15	Rvalue	651
6.1.5.2.3	Wselect	652
6.1.5.3	FormatPy	652
6.1.5.3.1	Dexport	653
6.1.5.3.1.1	Dseparator	653
6.1.5.3.1.2	Header	654
6.1.5.3.1.3	Traces	654
6.1.5.4	Initiate	655
6.1.5.4.1	ConMeas	655
6.1.5.4.2	Continuous	656
6.1.5.4.3	Immediate	657
6.1.5.5	InputPy<InputIx>	657
6.1.5.5.1	Attenuation	658
6.1.5.5.1.1	Auto	659
6.1.5.5.2	Connector	659
6.1.5.5.3	Coupling	660
6.1.5.5.4	Dpath	660
6.1.5.5.5	Egain	661
6.1.5.5.5.1	Bypass	661
6.1.5.5.5.2	State	662
6.1.5.5.6	FilterPy	663
6.1.5.5.6.1	Hpass	663
6.1.5.5.6.2	State	663
6.1.5.5.6.3	Yig	664
6.1.5.5.6.4	State	664
6.1.5.5.7	Gain	665
6.1.5.5.7.1	State	665
6.1.5.5.7.2	Value	666
6.1.5.5.8	Impedance	666
6.1.5.5.9	Select	667

6.1.5.6	Layout	668
6.1.5.6.1	Add	668
6.1.5.6.1.1	Window	668
6.1.5.6.2	Catalog	669
6.1.5.6.2.1	Window	669
6.1.5.6.3	Identify	670
6.1.5.6.3.1	Window	670
6.1.5.6.4	Remove	670
6.1.5.6.4.1	Window	671
6.1.5.6.5	Replace	671
6.1.5.6.5.1	Window	671
6.1.5.6.6	Splitter	672
6.1.5.7	MassMemory	673
6.1.5.7.1	Load	673
6.1.5.7.1.1	Iq	673
6.1.5.7.1.2	Stream	673
6.1.5.7.1.3	Auto	674
6.1.5.7.1.4	ListPy	675
6.1.5.7.2	Store<Store>	675
6.1.5.7.2.1	Trace	675
6.1.5.8	Output<OutputConnector>	676
6.1.5.8.1	Iqhs	676
6.1.5.8.1.1	Cdevice	677
6.1.5.8.1.2	SymbolRate	677
6.1.5.8.2	Trigger<TriggerPort>	678
6.1.5.8.2.1	Direction	678
6.1.5.8.2.2	Level	679
6.1.5.8.2.3	Otype	680
6.1.5.8.2.4	Pulse	680
6.1.5.8.2.5	Immediate	681
6.1.5.8.2.6	Length	681
6.1.5.9	Sense	682
6.1.5.9.1	Adjust	682
6.1.5.9.1.1	Level	682
6.1.5.9.2	Correction	683
6.1.5.9.2.1	Cvl	683
6.1.5.9.2.2	Band	684
6.1.5.9.2.3	Bias	685
6.1.5.9.2.4	Catalog	685
6.1.5.9.2.5	Comment	686
6.1.5.9.2.6	Data	686
6.1.5.9.2.7	Harmonic	687
6.1.5.9.2.8	Mixer	688
6.1.5.9.2.9	Ports	688
6.1.5.9.2.10	Select	689
6.1.5.9.2.11	Snumber	689
6.1.5.9.3	Frequency	690
6.1.5.9.3.1	Center	690
6.1.5.9.3.2	Step	691
6.1.5.9.3.3	Offset	692
6.1.5.9.4	Mixer	692
6.1.5.9.4.1	Bias	693
6.1.5.9.4.2	High	693
6.1.5.9.4.3	Low	694

6.1.5.9.4.4	Frequency	694
6.1.5.9.4.5	Handover	694
6.1.5.9.4.6	Start	695
6.1.5.9.4.7	Stop	695
6.1.5.9.4.8	Harmonic	696
6.1.5.9.4.9	Band	696
6.1.5.9.4.10	High	697
6.1.5.9.4.11	State	697
6.1.5.9.4.12	Value	698
6.1.5.9.4.13	Low	698
6.1.5.9.4.14	TypePy	699
6.1.5.9.4.15	LoPower	700
6.1.5.9.4.16	Loss	700
6.1.5.9.4.17	High	700
6.1.5.9.4.18	Low	701
6.1.5.9.4.19	Table	702
6.1.5.9.4.20	High	702
6.1.5.9.4.21	Low	703
6.1.5.9.4.22	Ports	703
6.1.5.9.4.23	RfOverrange	704
6.1.5.9.4.24	State	704
6.1.5.9.4.25	Signal	705
6.1.5.9.4.26	State	705
6.1.5.9.4.27	Threshold	706
6.1.5.9.5	Msra	706
6.1.5.9.5.1	Capture	707
6.1.5.9.5.2	Offset	707
6.1.5.9.6	Probe<Probe>	708
6.1.5.9.6.1	Id	708
6.1.5.9.6.2	PartNumber	708
6.1.5.9.6.3	SrNumber	709
6.1.5.9.6.4	Setup	709
6.1.5.9.6.5	AttRatio	709
6.1.5.9.6.6	CmOffset	710
6.1.5.9.6.7	DmOffset	711
6.1.5.9.6.8	Mode	712
6.1.5.9.6.9	Name	712
6.1.5.9.6.10	NmOffset	713
6.1.5.9.6.11	Pmode	714
6.1.5.9.6.12	PmOffset	714
6.1.5.9.6.13	State	715
6.1.5.9.6.14	TypePy	716
6.1.5.9.7	Sweep	716
6.1.5.9.7.1	Count	716
6.1.5.9.7.2	Current	717
6.1.5.10	Trace<Window>	718
6.1.5.10.1	Data	718
6.1.5.10.1.1	X	719
6.1.5.11	Trigger	719
6.1.5.11.1	Sequence	720
6.1.5.11.1.1	Dtime	720
6.1.5.11.1.2	Holdoff	721
6.1.5.11.1.3	Time	721
6.1.5.11.1.4	IfPower	721

6.1.5.11.1.5	Holdoff	722
6.1.5.11.1.6	Hysteresis	722
6.1.5.11.1.7	Level	723
6.1.5.11.1.8	External<ExternalPort>	723
6.1.5.11.1.9	IfPower	724
6.1.5.11.1.10	IqPower	725
6.1.5.11.1.11	RfPower	726
6.1.5.11.1.12	Slope	726
6.1.5.11.1.13	Source	727
6.1.5.12	TriggerInvoke	728
6.1.6	K7_AnalogDemod	728
6.1.6.1	Layout	729
6.1.6.1.1	Add	729
6.1.6.1.1.1	Window	729
6.1.6.1.2	Catalog	730
6.1.6.1.2.1	Window	730
6.1.6.1.3	Identify	731
6.1.6.1.3.1	Window	731
6.1.6.1.4	Replace	731
6.1.6.1.4.1	Window	732
6.1.7	K70_Vsa	732
6.1.7.1	Calculate<Window>	733
6.1.7.1.1	BitErrorRate	733
6.1.7.1.2	Ddem	734
6.1.7.1.2.1	Burst	734
6.1.7.1.2.2	Length	734
6.1.7.1.2.3	Spectrum	735
6.1.7.1.2.4	State	735
6.1.7.1.3	DeltaMarker<DeltaMarker>	736
6.1.7.1.3.1	Aoff	737
6.1.7.1.3.2	Maximum	737
6.1.7.1.3.3	Apeak	737
6.1.7.1.3.4	Left	738
6.1.7.1.3.5	Next	739
6.1.7.1.3.6	Peak	739
6.1.7.1.3.7	Right	740
6.1.7.1.3.8	Mburst	740
6.1.7.1.3.9	Minimum	741
6.1.7.1.3.10	Left	741
6.1.7.1.3.11	Next	742
6.1.7.1.3.12	Peak	742
6.1.7.1.3.13	Right	743
6.1.7.1.3.14	State	743
6.1.7.1.3.15	Trace	744
6.1.7.1.3.16	X	745
6.1.7.1.3.17	Absolute	745
6.1.7.1.3.18	Relative	746
6.1.7.1.3.19	Y	746
6.1.7.1.4	Dlabs	747
6.1.7.1.4.1	State	747
6.1.7.1.4.2	Value	748
6.1.7.1.5	DlRel	749
6.1.7.1.5.1	State	749
6.1.7.1.5.2	Value	750

6.1.7.1.6	Dsp	751
6.1.7.1.6.1	Result	751
6.1.7.1.6.2	Capture	751
6.1.7.1.6.3	Bursts	751
6.1.7.1.6.4	Patterns	752
6.1.7.1.6.5	Rrange	752
6.1.7.1.6.6	Current	753
6.1.7.1.6.7	Burst	753
6.1.7.1.6.8	Length	753
6.1.7.1.6.9	Present	754
6.1.7.1.6.10	Start	754
6.1.7.1.6.11	Pattern	755
6.1.7.1.6.12	Confidence	755
6.1.7.1.6.13	Correct	756
6.1.7.1.6.14	Present	756
6.1.7.1.6.15	Start	757
6.1.7.1.7	Elin	757
6.1.7.1.7.1	State	757
6.1.7.1.7.2	Value	759
6.1.7.1.8	Feed	759
6.1.7.1.9	FormatPy	760
6.1.7.1.10	Fsk	762
6.1.7.1.10.1	Deviation	762
6.1.7.1.10.2	Compensation	762
6.1.7.1.10.3	Reference	763
6.1.7.1.10.4	Relative	763
6.1.7.1.10.5	Value	764
6.1.7.1.11	Limit	765
6.1.7.1.11.1	Maccuracy	765
6.1.7.1.11.2	CfError	765
6.1.7.1.11.3	Current	766
6.1.7.1.11.4	Result	766
6.1.7.1.11.5	State	767
6.1.7.1.11.6	Value	767
6.1.7.1.11.7	Mean	768
6.1.7.1.11.8	Result	768
6.1.7.1.11.9	State	769
6.1.7.1.11.10	Value	770
6.1.7.1.11.11	Peak	770
6.1.7.1.11.12	Result	771
6.1.7.1.11.13	State	771
6.1.7.1.11.14	Value	772
6.1.7.1.11.15	Default	773
6.1.7.1.11.16	Evm	773
6.1.7.1.11.17	Pcurrent	773
6.1.7.1.11.18	Result	774
6.1.7.1.11.19	State	774
6.1.7.1.11.20	Value	775
6.1.7.1.11.21	Pmean	776
6.1.7.1.11.22	Result	776
6.1.7.1.11.23	State	777
6.1.7.1.11.24	Value	777
6.1.7.1.11.25	Ppeak	778
6.1.7.1.11.26	Result	778

6.1.7.1.11.27	State	779
6.1.7.1.11.28	Value	780
6.1.7.1.11.29	Rcurrent	780
6.1.7.1.11.30	Result	781
6.1.7.1.11.31	State	781
6.1.7.1.11.32	Value	782
6.1.7.1.11.33	Rmean	783
6.1.7.1.11.34	Result	783
6.1.7.1.11.35	State	784
6.1.7.1.11.36	Value	784
6.1.7.1.11.37	Rpeak	785
6.1.7.1.11.38	Result	785
6.1.7.1.11.39	State	786
6.1.7.1.11.40	Value	787
6.1.7.1.11.41	FdError	787
6.1.7.1.11.42	Current	788
6.1.7.1.11.43	Result	788
6.1.7.1.11.44	State	789
6.1.7.1.11.45	Value	789
6.1.7.1.11.46	Mean	790
6.1.7.1.11.47	Result	790
6.1.7.1.11.48	State	791
6.1.7.1.11.49	Value	792
6.1.7.1.11.50	Peak	792
6.1.7.1.11.51	Result	793
6.1.7.1.11.52	State	793
6.1.7.1.11.53	Value	794
6.1.7.1.11.54	FreqError	795
6.1.7.1.11.55	Pcurrent	795
6.1.7.1.11.56	Result	795
6.1.7.1.11.57	State	796
6.1.7.1.11.58	Value	797
6.1.7.1.11.59	Pmean	797
6.1.7.1.11.60	Result	798
6.1.7.1.11.61	State	798
6.1.7.1.11.62	Value	799
6.1.7.1.11.63	Ppeak	800
6.1.7.1.11.64	Result	800
6.1.7.1.11.65	State	801
6.1.7.1.11.66	Value	801
6.1.7.1.11.67	Rcurrent	802
6.1.7.1.11.68	Result	802
6.1.7.1.11.69	State	803
6.1.7.1.11.70	Value	804
6.1.7.1.11.71	Rmean	804
6.1.7.1.11.72	Result	805
6.1.7.1.11.73	State	805
6.1.7.1.11.74	Value	806
6.1.7.1.11.75	Rpeak	807
6.1.7.1.11.76	Result	807
6.1.7.1.11.77	State	808
6.1.7.1.11.78	Value	808
6.1.7.1.11.79	Merror	809
6.1.7.1.11.80	Pcurrent	809

6.1.7.1.11.81	Result	810
6.1.7.1.11.82	State	810
6.1.7.1.11.83	Value	811
6.1.7.1.11.84	Pmean	812
6.1.7.1.11.85	Result	812
6.1.7.1.11.86	State	813
6.1.7.1.11.87	Value	813
6.1.7.1.11.88	Ppeak	814
6.1.7.1.11.89	Result	814
6.1.7.1.11.90	State	815
6.1.7.1.11.91	Value	816
6.1.7.1.11.92	Rcurrent	816
6.1.7.1.11.93	Result	817
6.1.7.1.11.94	State	817
6.1.7.1.11.95	Value	818
6.1.7.1.11.96	Rmean	819
6.1.7.1.11.97	Result	819
6.1.7.1.11.98	State	820
6.1.7.1.11.99	Value	820
6.1.7.1.11.100	Rpeak	821
6.1.7.1.11.101	Result	821
6.1.7.1.11.102	State	822
6.1.7.1.11.103	Value	823
6.1.7.1.11.104	Ooffset	823
6.1.7.1.11.105	Current	824
6.1.7.1.11.106	Result	824
6.1.7.1.11.107	State	825
6.1.7.1.11.108	Value	825
6.1.7.1.11.109	Mean	826
6.1.7.1.11.110	Result	826
6.1.7.1.11.111	State	827
6.1.7.1.11.112	Value	828
6.1.7.1.11.113	Peak	828
6.1.7.1.11.114	Result	829
6.1.7.1.11.115	State	829
6.1.7.1.11.116	Value	830
6.1.7.1.11.117	Perror	831
6.1.7.1.11.118	Pcurrent	831
6.1.7.1.11.119	Result	831
6.1.7.1.11.120	State	832
6.1.7.1.11.121	Value	833
6.1.7.1.11.122	Pmean	833
6.1.7.1.11.123	Result	834
6.1.7.1.11.124	State	834
6.1.7.1.11.125	Value	835
6.1.7.1.11.126	Ppeak	836
6.1.7.1.11.127	Result	836
6.1.7.1.11.128	State	837
6.1.7.1.11.129	Value	837
6.1.7.1.11.130	Rcurrent	838
6.1.7.1.11.131	Result	838
6.1.7.1.11.132	State	839
6.1.7.1.11.133	Value	840
6.1.7.1.11.134	Rmean	840

6.1.7.1.11.135 Result	841
6.1.7.1.11.136 State	841
6.1.7.1.11.137 Value	842
6.1.7.1.11.138 Rpeak	843
6.1.7.1.11.139 Result	843
6.1.7.1.11.140 State	844
6.1.7.1.11.141 Value	844
6.1.7.1.11.142 Rho	845
6.1.7.1.11.143 Current	845
6.1.7.1.11.144 Result	846
6.1.7.1.11.145 State	846
6.1.7.1.11.146 Value	847
6.1.7.1.11.147 Mean	848
6.1.7.1.11.148 Result	848
6.1.7.1.11.149 State	849
6.1.7.1.11.150 Value	849
6.1.7.1.11.151 Peak	850
6.1.7.1.11.152 Result	850
6.1.7.1.11.153 State	851
6.1.7.1.11.154 Value	852
6.1.7.1.11.155 State	852
6.1.7.1.12 Marker<Marker>	853
6.1.7.1.12.1 Aoff	854
6.1.7.1.12.2 Function	854
6.1.7.1.12.3 Ddemod	854
6.1.7.1.12.4 Result	855
6.1.7.1.12.5 Statistic	855
6.1.7.1.12.6 Adroop	856
6.1.7.1.12.7 All	857
6.1.7.1.12.8 CfError	857
6.1.7.1.12.9 Evm	858
6.1.7.1.12.10 FdError	859
6.1.7.1.12.11 Fsk	859
6.1.7.1.12.12 Cfdrift	860
6.1.7.1.12.13 Derror	860
6.1.7.1.12.14 Mdeviation	861
6.1.7.1.12.15 Rdeviation	862
6.1.7.1.12.16 Gimbalance	862
6.1.7.1.12.17 IqImbalance	863
6.1.7.1.12.18 Merror	864
6.1.7.1.12.19 Mpower	864
6.1.7.1.12.20 Ooffset	865
6.1.7.1.12.21 Perror	866
6.1.7.1.12.22 Qerror	866
6.1.7.1.12.23 Rho	867
6.1.7.1.12.24 Snr	868
6.1.7.1.12.25 SrError	868
6.1.7.1.12.26 Link	869
6.1.7.1.12.27 Maximum	870
6.1.7.1.12.28 Apeak	870
6.1.7.1.12.29 Left	871
6.1.7.1.12.30 Next	871
6.1.7.1.12.31 Peak	872
6.1.7.1.12.32 Right	872

6.1.7.1.12.33	Mburst	873
6.1.7.1.12.34	Minimum	873
6.1.7.1.12.35	Left	874
6.1.7.1.12.36	Next	874
6.1.7.1.12.37	Peak	875
6.1.7.1.12.38	Right	875
6.1.7.1.12.39	Search	876
6.1.7.1.12.40	State	877
6.1.7.1.12.41	Trace	878
6.1.7.1.12.42	X	879
6.1.7.1.12.43	Slimits	879
6.1.7.1.12.44	Left	879
6.1.7.1.12.45	Right	880
6.1.7.1.12.46	State	881
6.1.7.1.12.47	Y	882
6.1.7.1.13	Meas	882
6.1.7.1.13.1	Dirty	883
6.1.7.1.14	Msra	883
6.1.7.1.14.1	Aline	883
6.1.7.1.14.2	Show	884
6.1.7.1.14.3	Value	885
6.1.7.1.14.4	Window	885
6.1.7.1.14.5	Ival	886
6.1.7.1.15	Statistics	886
6.1.7.1.15.1	CumulativeDistribFnc	887
6.1.7.1.15.2	State	887
6.1.7.1.15.3	Mode	888
6.1.7.1.15.4	Scale	889
6.1.7.1.15.5	X	889
6.1.7.1.15.6	Bcount	889
6.1.7.1.15.7	Y	890
6.1.7.1.15.8	Lower	890
6.1.7.1.15.9	Unit	891
6.1.7.1.15.10	Upper	892
6.1.7.1.16	TlAbs	892
6.1.7.1.16.1	State	893
6.1.7.1.16.2	Value	893
6.1.7.1.17	TlRel	894
6.1.7.1.17.1	State	894
6.1.7.1.17.2	Value	895
6.1.7.1.18	Trace<Trace>	896
6.1.7.1.18.1	Adjust	896
6.1.7.1.18.2	Alignment	897
6.1.7.1.18.3	Default	897
6.1.7.1.18.4	Offset	898
6.1.7.1.18.5	Value	899
6.1.7.1.18.6	Symbols	900
6.1.7.1.18.7	Value	901
6.1.7.1.19	Unit	902
6.1.7.1.19.1	Angle	902
6.1.7.1.19.2	Power	903
6.1.7.1.20	X	903
6.1.7.1.20.1	Unit	904
6.1.7.1.20.2	Time	904

6.1.7.1.21	Y	905
6.1.7.1.21.1	Unit	905
6.1.7.1.21.2	Time	905
6.1.7.2	Display	906
6.1.7.2.1	Window<Window>	906
6.1.7.2.1.1	Item	907
6.1.7.2.1.2	Line	907
6.1.7.2.1.3	Value	907
6.1.7.2.1.4	Prate	908
6.1.7.2.1.5	Auto	909
6.1.7.2.1.6	Value	909
6.1.7.2.1.7	Size	910
6.1.7.2.1.8	State	911
6.1.7.2.1.9	Trace<Trace>	912
6.1.7.2.1.10	Mode	912
6.1.7.2.1.11	Preset	913
6.1.7.2.1.12	State	914
6.1.7.2.1.13	Symbol	915
6.1.7.2.1.14	X	916
6.1.7.2.1.15	Scale	916
6.1.7.2.1.16	Pdivision	916
6.1.7.2.1.17	RefPosition	917
6.1.7.2.1.18	Rvalue	918
6.1.7.2.1.19	Start	919
6.1.7.2.1.20	Stop	920
6.1.7.2.1.21	Voffset	920
6.1.7.2.1.22	Y	921
6.1.7.2.1.23	Scale	921
6.1.7.2.1.24	Auto	922
6.1.7.2.1.25	All	922
6.1.7.2.1.26	Mode	922
6.1.7.2.1.27	Pdivision	923
6.1.7.2.1.28	RefLevel	924
6.1.7.2.1.29	Offset	924
6.1.7.2.1.30	RefPosition	925
6.1.7.2.1.31	Rvalue	926
6.1.7.2.1.32	Spacing	927
6.1.7.3	FormatPy	928
6.1.7.3.1	Dexport	928
6.1.7.3.1.1	Dseparator	928
6.1.7.3.1.2	Header	929
6.1.7.3.1.3	Mode	930
6.1.7.4	Initiate	930
6.1.7.4.1	ConMeas	930
6.1.7.4.2	Continuous	931
6.1.7.4.3	Immediate	932
6.1.7.4.4	RefMeas	933
6.1.7.4.5	Refresh	933
6.1.7.5	InputPy<InputIx>	934
6.1.7.5.1	Attenuation	934
6.1.7.5.1.1	Auto	935
6.1.7.5.1.2	Mode	935
6.1.7.5.2	Coupling	936
6.1.7.5.3	Dpath	936

6.1.7.5.4	FilterPy	937
6.1.7.5.4.1	Hpass	937
6.1.7.5.4.2	State	938
6.1.7.5.4.3	Yig	939
6.1.7.5.4.4	State	939
6.1.7.5.5	Gain	940
6.1.7.5.5.1	State	940
6.1.7.5.5.2	Value	941
6.1.7.5.6	Select	942
6.1.7.6	Layout	943
6.1.7.6.1	Add	943
6.1.7.6.1.1	Window	943
6.1.7.6.2	Catalog	944
6.1.7.6.2.1	Window	944
6.1.7.6.3	Identify	945
6.1.7.6.3.1	Window	945
6.1.7.6.4	Move	945
6.1.7.6.4.1	Window	946
6.1.7.6.5	Remove	946
6.1.7.6.5.1	Window	947
6.1.7.6.6	Replace	947
6.1.7.6.6.1	Window	947
6.1.7.6.7	Splitter	948
6.1.7.7	MassMemory	949
6.1.7.7.1	Load<Window>	949
6.1.7.7.1.1	Iq	949
6.1.7.7.1.2	State	950
6.1.7.7.1.3	Stream	950
6.1.7.7.1.4	Auto	951
6.1.7.7.1.5	ListPy	951
6.1.7.7.2	Store<Store>	952
6.1.7.7.2.1	Iq	952
6.1.7.7.2.2	Comment	952
6.1.7.7.2.3	FormatPy	953
6.1.7.7.2.4	State	954
6.1.7.7.2.5	Trace	954
6.1.7.7.2.6	Trace	955
6.1.7.8	Output<OutputConnector>	956
6.1.7.8.1	Ifreq	956
6.1.7.8.1.1	Source	956
6.1.7.8.2	Trigger<TriggerPort>	957
6.1.7.8.2.1	Direction	958
6.1.7.8.2.2	Level	959
6.1.7.8.2.3	Otype	960
6.1.7.8.2.4	Pulse	961
6.1.7.8.2.5	Immediate	961
6.1.7.8.2.6	Length	961
6.1.7.9	Sense	962
6.1.7.9.1	Adjust	963
6.1.7.9.1.1	Configure	963
6.1.7.9.1.2	Hysteresis	963
6.1.7.9.1.3	Lower	963
6.1.7.9.1.4	Upper	964
6.1.7.9.1.5	Level	965

6.1.7.9.1.6	Duration	965
6.1.7.9.1.7	Mode	965
6.1.7.9.1.8	Level	966
6.1.7.9.2	Ddemod	967
6.1.7.9.2.1	Apsk	967
6.1.7.9.2.2	Nstate	967
6.1.7.9.2.3	Ask	968
6.1.7.9.2.4	Nstate	968
6.1.7.9.2.5	Bordering	969
6.1.7.9.2.6	Ecalc	969
6.1.7.9.2.7	Mode	970
6.1.7.9.2.8	Offset	970
6.1.7.9.2.9	EpRate	971
6.1.7.9.2.10	Auto	971
6.1.7.9.2.11	Value	972
6.1.7.9.2.12	Equalizer	973
6.1.7.9.2.13	File	974
6.1.7.9.2.14	FormatPy	974
6.1.7.9.2.15	Length	975
6.1.7.9.2.16	Load	975
6.1.7.9.2.17	Mode	976
6.1.7.9.2.18	Save	977
6.1.7.9.2.19	State	977
6.1.7.9.2.20	Factory	978
6.1.7.9.2.21	Value	978
6.1.7.9.2.22	FilterPy	978
6.1.7.9.2.23	Alpha	979
6.1.7.9.2.24	Reference	979
6.1.7.9.2.25	State	980
6.1.7.9.2.26	FormatPy	981
6.1.7.9.2.27	Fsk	981
6.1.7.9.2.28	Nstate	982
6.1.7.9.2.29	Fsync	982
6.1.7.9.2.30	Auto	982
6.1.7.9.2.31	Level	983
6.1.7.9.2.32	Mode	984
6.1.7.9.2.33	Result	985
6.1.7.9.2.34	Kdata	985
6.1.7.9.2.35	Name	985
6.1.7.9.2.36	Ser	986
6.1.7.9.2.37	Limit	986
6.1.7.9.2.38	State	987
6.1.7.9.2.39	Mapping	987
6.1.7.9.2.40	Catalog	987
6.1.7.9.2.41	Value	988
6.1.7.9.2.42	Mfilter	989
6.1.7.9.2.43	Alpha	989
6.1.7.9.2.44	Auto	989
6.1.7.9.2.45	Name	990
6.1.7.9.2.46	State	991
6.1.7.9.2.47	User	991
6.1.7.9.2.48	Msk	992
6.1.7.9.2.49	FormatPy	992
6.1.7.9.2.50	Normalize	993

6.1.7.9.2.51	Adroop	993
6.1.7.9.2.52	Cfdrift	994
6.1.7.9.2.53	Channel	994
6.1.7.9.2.54	FdError	995
6.1.7.9.2.55	IqImbalance	995
6.1.7.9.2.56	IqOffset	996
6.1.7.9.2.57	SrError	996
6.1.7.9.2.58	Value	997
6.1.7.9.2.59	Optimization	998
6.1.7.9.2.60	Pattern	998
6.1.7.9.2.61	Apsk	999
6.1.7.9.2.62	Nstate	999
6.1.7.9.2.63	Ask	1000
6.1.7.9.2.64	Nstate	1000
6.1.7.9.2.65	FormatPy	1001
6.1.7.9.2.66	Frame	1002
6.1.7.9.2.67	Edit	1002
6.1.7.9.2.68	Next	1003
6.1.7.9.2.69	Boosting	1004
6.1.7.9.2.70	Modulation	1004
6.1.7.9.2.71	Previous	1005
6.1.7.9.2.72	Boosting	1005
6.1.7.9.2.73	Modulation	1006
6.1.7.9.2.74	Structure	1007
6.1.7.9.2.75	Text	1009
6.1.7.9.2.76	Load	1009
6.1.7.9.2.77	Mode	1010
6.1.7.9.2.78	Mapping	1011
6.1.7.9.2.79	Catalog	1011
6.1.7.9.2.80	Value	1012
6.1.7.9.2.81	Psk	1012
6.1.7.9.2.82	FormatPy	1013
6.1.7.9.2.83	Nstate	1014
6.1.7.9.2.84	Qam	1015
6.1.7.9.2.85	FormatPy	1015
6.1.7.9.2.86	Nstate	1016
6.1.7.9.2.87	Qpsk	1017
6.1.7.9.2.88	FormatPy	1017
6.1.7.9.2.89	State	1018
6.1.7.9.2.90	User	1018
6.1.7.9.2.91	Name	1019
6.1.7.9.2.92	Prate	1019
6.1.7.9.2.93	Preset	1020
6.1.7.9.2.94	Calc	1020
6.1.7.9.2.95	RefLevel	1021
6.1.7.9.2.96	Standard	1022
6.1.7.9.2.97	Psk	1023
6.1.7.9.2.98	FormatPy	1023
6.1.7.9.2.99	Nstate	1024
6.1.7.9.2.100	Qam	1025
6.1.7.9.2.101	FormatPy	1025
6.1.7.9.2.102	Nstate	1026
6.1.7.9.2.103	Qpsk	1027
6.1.7.9.2.104	FormatPy	1027

6.1.7.9.2.105	Rlength	1028
6.1.7.9.2.106	Auto	1028
6.1.7.9.2.107	Symbols	1029
6.1.7.9.2.108	Value	1029
6.1.7.9.2.109	Value	1030
6.1.7.9.2.110	Sband	1031
6.1.7.9.2.111	Search	1031
6.1.7.9.2.112	Burst	1032
6.1.7.9.2.113	Auto	1032
6.1.7.9.2.114	Configure	1033
6.1.7.9.2.115	Auto	1033
6.1.7.9.2.116	Glength	1034
6.1.7.9.2.117	Minimum	1034
6.1.7.9.2.118	Length	1035
6.1.7.9.2.119	Maximum	1035
6.1.7.9.2.120	Minimum	1036
6.1.7.9.2.121	Mode	1036
6.1.7.9.2.122	Skip	1037
6.1.7.9.2.123	Falling	1037
6.1.7.9.2.124	Rising	1038
6.1.7.9.2.125	State	1038
6.1.7.9.2.126	Tolerance	1039
6.1.7.9.2.127	Mburst	1040
6.1.7.9.2.128	Calc	1040
6.1.7.9.2.129	Start	1040
6.1.7.9.2.130	Samples	1041
6.1.7.9.2.131	Symbols	1041
6.1.7.9.2.132	Pattern	1042
6.1.7.9.2.133	Configure	1042
6.1.7.9.2.134	Auto	1042
6.1.7.9.2.135	Sync	1043
6.1.7.9.2.136	Auto	1043
6.1.7.9.2.137	State	1044
6.1.7.9.2.138	Pulse	1044
6.1.7.9.2.139	State	1045
6.1.7.9.2.140	Sync	1045
6.1.7.9.2.141	Auto	1046
6.1.7.9.2.142	Catalog	1047
6.1.7.9.2.143	Comment	1048
6.1.7.9.2.144	Data	1048
6.1.7.9.2.145	Found	1049
6.1.7.9.2.146	IqcThreshold	1049
6.1.7.9.2.147	Mode	1050
6.1.7.9.2.148	Name	1051
6.1.7.9.2.149	Nstate	1051
6.1.7.9.2.150	Offset	1052
6.1.7.9.2.151	Pattern	1052
6.1.7.9.2.152	Add	1053
6.1.7.9.2.153	Remove	1054
6.1.7.9.2.154	Select	1054
6.1.7.9.2.155	State	1055
6.1.7.9.2.156	Text	1055
6.1.7.9.2.157	Time	1056
6.1.7.9.2.158	Signal	1056

6.1.7.9.2.159	Pattern	1057
6.1.7.9.2.160	Value	1057
6.1.7.9.2.161	Standard	1058
6.1.7.9.2.162	Comment	1059
6.1.7.9.2.163	Preset	1059
6.1.7.9.2.164	Value	1060
6.1.7.9.2.165	Sync	1060
6.1.7.9.2.166	Offset	1060
6.1.7.9.2.167	State	1061
6.1.7.9.2.168	Value	1061
6.1.7.9.2.169	SymbolRate	1062
6.1.7.9.2.170	Tfilter	1063
6.1.7.9.2.171	Alpha	1063
6.1.7.9.2.172	Name	1063
6.1.7.9.2.173	State	1064
6.1.7.9.2.174	User	1065
6.1.7.9.2.175	Time	1065
6.1.7.9.2.176	Auto	1066
6.1.7.9.2.177	User	1067
6.1.7.9.2.178	Name	1067
6.1.7.9.3	Frequency	1068
6.1.7.9.3.1	Center	1068
6.1.7.9.3.2	Step	1068
6.1.7.9.3.3	Auto	1068
6.1.7.9.3.4	Offset	1069
6.1.7.9.4	Msra	1070
6.1.7.9.4.1	Capture	1070
6.1.7.9.4.2	Offset	1070
6.1.7.9.5	SwapIq	1071
6.1.7.9.6	Sweep	1072
6.1.7.9.6.1	Count	1072
6.1.7.9.6.2	Current	1072
6.1.7.9.6.3	Value	1073
6.1.7.9.7	Tcapture	1073
6.1.7.9.7.1	Length	1073
6.1.7.10	Status	1074
6.1.7.10.1	Questionable	1074
6.1.7.10.1.1	Modulation<Window>	1075
6.1.7.10.1.2	Cfrequency	1075
6.1.7.10.1.3	Condition	1075
6.1.7.10.1.4	Enable	1076
6.1.7.10.1.5	Event	1077
6.1.7.10.1.6	Ntransition	1077
6.1.7.10.1.7	Ptransition	1078
6.1.7.10.1.8	Condition	1079
6.1.7.10.1.9	Enable	1080
6.1.7.10.1.10	Event	1081
6.1.7.10.1.11	Evm	1081
6.1.7.10.1.12	Condition	1082
6.1.7.10.1.13	Enable	1082
6.1.7.10.1.14	Event	1083
6.1.7.10.1.15	Ntransition	1084
6.1.7.10.1.16	Ptransition	1085
6.1.7.10.1.17	Fsk	1086

6.1.7.10.1.18	Condition	1086
6.1.7.10.1.19	Enable	1086
6.1.7.10.1.20	Event	1087
6.1.7.10.1.21	Ntransition	1088
6.1.7.10.1.22	Ptransition	1089
6.1.7.10.1.23	IqRho	1090
6.1.7.10.1.24	Condition	1090
6.1.7.10.1.25	Enable	1091
6.1.7.10.1.26	Event	1092
6.1.7.10.1.27	Ntransition	1092
6.1.7.10.1.28	Ptransition	1093
6.1.7.10.1.29	Magnitude	1094
6.1.7.10.1.30	Condition	1094
6.1.7.10.1.31	Enable	1095
6.1.7.10.1.32	Event	1096
6.1.7.10.1.33	Ntransition	1096
6.1.7.10.1.34	Ptransition	1097
6.1.7.10.1.35	Ntransition	1098
6.1.7.10.1.36	Phase	1099
6.1.7.10.1.37	Condition	1100
6.1.7.10.1.38	Enable	1100
6.1.7.10.1.39	Event	1101
6.1.7.10.1.40	Ntransition	1102
6.1.7.10.1.41	Ptransition	1103
6.1.7.10.1.42	Ptransition	1104
6.1.7.11	Trace<Window>	1105
6.1.7.11.1	Data	1105
6.1.7.11.2	Iq	1106
6.1.7.11.2.1	Bandwidth	1106
6.1.7.11.2.2	File	1106
6.1.7.11.2.3	Repetition	1107
6.1.7.11.2.4	Count	1107
6.1.7.11.2.5	Wband	1108
6.1.7.11.2.6	Mbwidth	1108
6.1.7.11.2.7	State	1109
6.1.7.12	Trigger<TriggerPort>	1109
6.1.7.12.1	Sequence	1110
6.1.7.12.1.1	Dtime	1110
6.1.7.12.1.2	Holdoff	1111
6.1.7.12.1.3	Time	1111
6.1.7.12.1.4	IfPower	1112
6.1.7.12.1.5	Holdoff	1112
6.1.7.12.1.6	Hysteresis	1113
6.1.7.12.1.7	Level	1114
6.1.7.12.1.8	External<ExternalPort>	1114
6.1.7.12.1.9	IfPower	1115
6.1.7.12.1.10	IqPower	1116
6.1.7.12.1.11	RfPower	1117
6.1.7.12.1.12	Video	1117
6.1.7.12.1.13	RfPower	1118
6.1.7.12.1.14	Holdoff	1118
6.1.7.12.1.15	Slope	1119
6.1.7.12.1.16	Time	1120
6.1.7.12.1.17	Rinterval	1120

6.2	Calculate<Window>	1121
6.2.1	DeltaMarker<DeltaMarker>	1121
6.2.1.1	Aoff	1122
6.2.1.2	Function	1122
6.2.1.2.1	AfPhase	1122
6.2.1.2.1.1	Result	1123
6.2.1.2.1.2	State	1123
6.2.1.2.2	Bpower	1124
6.2.1.2.2.1	Mode	1124
6.2.1.2.2.2	Result	1125
6.2.1.2.2.3	Span	1126
6.2.1.2.2.4	State	1127
6.2.1.2.3	Fixed	1128
6.2.1.2.3.1	Rpoint	1128
6.2.1.2.3.2	Maximum	1128
6.2.1.2.3.3	Peak	1128
6.2.1.2.3.4	X	1129
6.2.1.2.3.5	Y	1130
6.2.1.2.3.6	Offset	1131
6.2.1.2.3.7	State	1132
6.2.1.2.4	Pnoise	1133
6.2.1.2.4.1	Auto	1133
6.2.1.2.4.2	Result	1133
6.2.1.2.4.3	State	1134
6.2.1.3	LinkTo	1135
6.2.1.3.1	Marker<MarkerDestination>	1135
6.2.1.4	Maximum	1136
6.2.1.4.1	Left	1136
6.2.1.4.2	Next	1137
6.2.1.4.3	Peak	1137
6.2.1.4.4	Right	1138
6.2.1.5	Minimum	1138
6.2.1.5.1	Left	1139
6.2.1.5.2	Next	1139
6.2.1.5.3	Peak	1140
6.2.1.5.4	Right	1140
6.2.1.6	Mode	1141
6.2.1.7	Mref	1142
6.2.1.8	Mreference	1143
6.2.1.9	Spectrogram	1143
6.2.1.9.1	Frame	1144
6.2.1.9.2	Sarea	1145
6.2.1.9.3	Xy	1146
6.2.1.9.3.1	Maximum	1146
6.2.1.9.3.2	Peak	1146
6.2.1.9.3.3	Minimum	1147
6.2.1.9.3.4	Peak	1147
6.2.1.9.4	Y	1148
6.2.1.9.4.1	Maximum	1148
6.2.1.9.4.2	Above	1148
6.2.1.9.4.3	Below	1149
6.2.1.9.4.4	Next	1149
6.2.1.9.4.5	Peak	1150
6.2.1.9.4.6	Minimum	1150

	6.2.1.9.4.7	Above	1151
	6.2.1.9.4.8	Below	1151
	6.2.1.9.4.9	Next	1152
	6.2.1.9.4.10	Peak	1152
6.2.1.10	State		1153
6.2.1.11	Trace		1154
6.2.1.12	X		1155
	6.2.1.12.1	Relative	1156
6.2.1.13	Y		1156
6.2.2	Espectrum		1157
6.2.2.1	PeakSearch		1157
	6.2.2.1.1	Auto	1158
	6.2.2.1.2	Immediate	1158
	6.2.2.1.3	Margin	1159
	6.2.2.1.4	Pshow	1159
6.2.3	Limit<LimitIx>		1160
6.2.3.1	AcPower		1161
	6.2.3.1.1	Achannel	1161
	6.2.3.1.1.1	Absolute	1161
	6.2.3.1.1.2	State	1162
	6.2.3.1.1.3	Relative	1164
	6.2.3.1.1.4	State	1165
	6.2.3.1.1.5	Result	1166
	6.2.3.1.1.6	Absolute	1167
	6.2.3.1.1.7	Relative	1167
	6.2.3.1.2	Alternate<Channel>	1168
	6.2.3.1.2.1	Absolute	1168
	6.2.3.1.2.2	State	1170
	6.2.3.1.2.3	Relative	1171
	6.2.3.1.2.4	State	1172
	6.2.3.1.2.5	Result	1174
	6.2.3.1.2.6	Absolute	1174
	6.2.3.1.2.7	Relative	1175
	6.2.3.1.3	Gap<GapChannel>	1176
	6.2.3.1.3.1	Aclr	1176
	6.2.3.1.3.2	Result	1177
	6.2.3.1.3.3	Auto	1177
	6.2.3.1.3.4	Absolute	1178
	6.2.3.1.3.5	State	1179
	6.2.3.1.3.6	Aclr	1180
	6.2.3.1.3.7	Relative	1180
	6.2.3.1.3.8	State	1182
	6.2.3.1.3.9	Caclr	1183
	6.2.3.1.3.10	Relative	1183
	6.2.3.1.3.11	State	1184
	6.2.3.1.3.12	Caclr	1185
	6.2.3.1.3.13	Result	1186
	6.2.3.1.3.14	Absolute	1187
	6.2.3.1.3.15	Relative	1188
	6.2.3.1.3.16	Manual	1189
	6.2.3.1.3.17	Lower	1189
	6.2.3.1.3.18	Absolute	1189
	6.2.3.1.3.19	State	1190
	6.2.3.1.3.20	Aclr	1192

6.2.3.1.3.21	Relative	1192
6.2.3.1.3.22	State	1193
6.2.3.1.3.23	Caclr	1194
6.2.3.1.3.24	Relative	1195
6.2.3.1.3.25	State	1196
6.2.3.1.3.26	Upper	1197
6.2.3.1.3.27	Absolute	1198
6.2.3.1.3.28	State	1199
6.2.3.1.3.29	Aclr	1200
6.2.3.1.3.30	Relative	1200
6.2.3.1.3.31	State	1202
6.2.3.1.3.32	Caclr	1203
6.2.3.1.3.33	Relative	1203
6.2.3.1.3.34	State	1205
6.2.3.1.4	State	1206
6.2.3.2	Active	1207
6.2.3.3	Clear	1207
6.2.3.3.1	Immediate	1207
6.2.3.4	Comment	1208
6.2.3.5	Control	1209
6.2.3.5.1	Data	1209
6.2.3.5.2	Domain	1210
6.2.3.5.3	Mode	1211
6.2.3.5.4	Offset	1212
6.2.3.5.5	Shift	1213
6.2.3.5.6	Spacing	1214
6.2.3.6	Copy	1215
6.2.3.7	Espectrum<SubBlock>	1216
6.2.3.7.1	Limits	1216
6.2.3.7.2	Mode	1217
6.2.3.7.3	Pclass<PowerClass>	1218
6.2.3.7.3.1	Count	1219
6.2.3.7.3.2	Exclusive	1220
6.2.3.7.3.3	Limit	1221
6.2.3.7.3.4	State	1221
6.2.3.7.3.5	Maximum	1223
6.2.3.7.3.6	Minimum	1224
6.2.3.7.4	Restore	1225
6.2.3.7.5	Value	1226
6.2.3.8	Fail	1227
6.2.3.9	Lower	1227
6.2.3.9.1	Data	1227
6.2.3.9.2	Margin	1228
6.2.3.9.3	Mode	1229
6.2.3.9.4	Offset	1230
6.2.3.9.5	Shift	1231
6.2.3.9.6	Spacing	1232
6.2.3.9.7	State	1233
6.2.3.9.8	Threshold	1234
6.2.3.10	Name	1235
6.2.3.11	State	1235
6.2.3.12	Trace<Trace>	1236
6.2.3.12.1	Check	1238
6.2.3.13	Unit	1239

6.2.3.14	Upper	1240
6.2.3.14.1	Data	1240
6.2.3.14.2	Margin	1241
6.2.3.14.3	Mode	1242
6.2.3.14.4	Offset	1243
6.2.3.14.5	Shift	1244
6.2.3.14.6	Spacing	1245
6.2.3.14.7	State	1246
6.2.3.14.8	Threshold	1247
6.2.4	Marker<Marker>	1248
6.2.4.1	Aoff	1248
6.2.4.2	Count	1249
6.2.4.2.1	Frequency	1250
6.2.4.2.2	Resolution	1250
6.2.4.3	Function	1251
6.2.4.3.1	Ademod	1251
6.2.4.3.1.1	Afrequency	1252
6.2.4.3.1.2	Result<Trace>	1252
6.2.4.3.1.3	Am	1253
6.2.4.3.1.4	Result<Trace>	1253
6.2.4.3.1.5	Relative	1254
6.2.4.3.1.6	Carrier	1255
6.2.4.3.1.7	Result<Trace>	1255
6.2.4.3.1.8	Distortion	1256
6.2.4.3.1.9	Write	1256
6.2.4.3.1.10	Result<Trace>	1257
6.2.4.3.1.11	Fm	1258
6.2.4.3.1.12	Result<Trace>	1258
6.2.4.3.1.13	Relative	1259
6.2.4.3.1.14	FreqError	1260
6.2.4.3.1.15	Result<Trace>	1260
6.2.4.3.1.16	Pm	1261
6.2.4.3.1.17	Result<Trace>	1261
6.2.4.3.1.18	Relative	1262
6.2.4.3.1.19	Sinad	1263
6.2.4.3.1.20	Result<Trace>	1263
6.2.4.3.1.21	Thd	1264
6.2.4.3.1.22	Result<Trace>	1265
6.2.4.3.2	AfPhase	1266
6.2.4.3.2.1	Result	1266
6.2.4.3.2.2	State	1267
6.2.4.3.3	Bpower	1268
6.2.4.3.3.1	Aoff	1268
6.2.4.3.3.2	Mode	1268
6.2.4.3.3.3	Result	1269
6.2.4.3.3.4	Span	1270
6.2.4.3.3.5	State	1271
6.2.4.3.4	Center	1272
6.2.4.3.5	Cstep	1272
6.2.4.3.6	Fpeaks	1273
6.2.4.3.6.1	Annotation	1273
6.2.4.3.6.2	Label	1273
6.2.4.3.6.3	State	1273
6.2.4.3.6.4	Count	1274

6.2.4.3.6.5	Immediate	1275
6.2.4.3.6.6	ListPy	1275
6.2.4.3.6.7	Size	1276
6.2.4.3.6.8	Sort	1277
6.2.4.3.6.9	State	1278
6.2.4.3.6.10	X	1279
6.2.4.3.6.11	Y	1279
6.2.4.3.7	Harmonics	1280
6.2.4.3.7.1	Bandwidth	1280
6.2.4.3.7.2	Auto	1281
6.2.4.3.7.3	Distortion	1281
6.2.4.3.7.4	ListPy	1282
6.2.4.3.7.5	Nharmonics	1283
6.2.4.3.7.6	State	1284
6.2.4.3.8	Mdepth	1285
6.2.4.3.8.1	Result<Trace>	1285
6.2.4.3.8.2	State	1286
6.2.4.3.9	Msummary	1287
6.2.4.3.10	NdbDown	1288
6.2.4.3.10.1	Frequency	1289
6.2.4.3.10.2	Qfactor	1289
6.2.4.3.10.3	Result	1290
6.2.4.3.10.4	State	1290
6.2.4.3.10.5	Time	1291
6.2.4.3.11	Noise	1292
6.2.4.3.11.1	Aoff	1292
6.2.4.3.11.2	Result	1293
6.2.4.3.11.3	State	1293
6.2.4.3.12	Pnoise	1294
6.2.4.3.12.1	Aoff	1294
6.2.4.3.12.2	Result	1295
6.2.4.3.12.3	State	1295
6.2.4.3.13	Power<SubBlock>	1296
6.2.4.3.13.1	Auto	1297
6.2.4.3.13.2	ListPy	1297
6.2.4.3.13.3	Mode	1298
6.2.4.3.13.4	Result	1299
6.2.4.3.13.5	Details	1300
6.2.4.3.13.6	Phz	1301
6.2.4.3.13.7	Unit	1302
6.2.4.3.13.8	Select	1303
6.2.4.3.13.9	Standard	1304
6.2.4.3.13.10	Catalog	1304
6.2.4.3.13.11	Save	1305
6.2.4.3.13.12	State	1306
6.2.4.3.14	Reference	1307
6.2.4.3.15	Strack	1308
6.2.4.3.15.1	State	1308
6.2.4.3.15.2	Threshold	1309
6.2.4.3.15.3	Trace	1310
6.2.4.3.16	Summary	1311
6.2.4.3.16.1	Aoff	1311
6.2.4.3.16.2	Average	1311
6.2.4.3.16.3	Mean	1312

6.2.4.3.16.4	Average	1313
6.2.4.3.16.5	Result	1313
6.2.4.3.16.6	Phold	1313
6.2.4.3.16.7	Result	1314
6.2.4.3.16.8	Result	1314
6.2.4.3.16.9	State	1315
6.2.4.3.16.10	Phold	1316
6.2.4.3.16.11	Ppeak	1317
6.2.4.3.16.12	Average	1317
6.2.4.3.16.13	Result	1317
6.2.4.3.16.14	Phold	1318
6.2.4.3.16.15	Result	1318
6.2.4.3.16.16	Result	1319
6.2.4.3.16.17	State	1319
6.2.4.3.16.18	Rms	1320
6.2.4.3.16.19	Average	1320
6.2.4.3.16.20	Result	1321
6.2.4.3.16.21	Phold	1321
6.2.4.3.16.22	Result	1321
6.2.4.3.16.23	Result	1322
6.2.4.3.16.24	State	1323
6.2.4.3.16.25	StandardDev	1324
6.2.4.3.16.26	Average	1324
6.2.4.3.16.27	Result	1324
6.2.4.3.16.28	Phold	1325
6.2.4.3.16.29	Result	1325
6.2.4.3.16.30	Result	1326
6.2.4.3.16.31	State	1326
6.2.4.3.16.32	State	1327
6.2.4.3.17	Toi	1328
6.2.4.3.17.1	Result	1328
6.2.4.3.17.2	Maximum	1329
6.2.4.3.17.3	Minimum	1329
6.2.4.3.17.4	State	1330
6.2.4.4	Link	1331
6.2.4.5	LinkTo	1331
6.2.4.5.1	Marker<MarkerDestination>	1332
6.2.4.6	LoExclude	1333
6.2.4.7	Maximum	1334
6.2.4.7.1	Auto	1334
6.2.4.7.2	Left	1334
6.2.4.7.3	Next	1335
6.2.4.7.4	Peak	1336
6.2.4.7.5	Right	1336
6.2.4.8	Minimum	1337
6.2.4.8.1	Auto	1337
6.2.4.8.2	Left	1338
6.2.4.8.3	Next	1338
6.2.4.8.4	Peak	1339
6.2.4.8.5	Right	1339
6.2.4.9	Pexcursion	1340
6.2.4.10	Spectrogram	1341
6.2.4.10.1	Frame	1341
6.2.4.10.2	Sarea	1342

6.2.4.10.3	Xy	1343
6.2.4.10.3.1	Maximum	1343
6.2.4.10.3.2	Peak	1343
6.2.4.10.3.3	Minimum	1344
6.2.4.10.3.4	Peak	1344
6.2.4.10.4	Y	1344
6.2.4.10.4.1	Maximum	1345
6.2.4.10.4.2	Above	1345
6.2.4.10.4.3	Below	1346
6.2.4.10.4.4	Next	1346
6.2.4.10.4.5	Peak	1347
6.2.4.10.4.6	Minimum	1347
6.2.4.10.4.7	Above	1347
6.2.4.10.4.8	Below	1348
6.2.4.10.4.9	Next	1348
6.2.4.10.4.10	Peak	1349
6.2.4.11	State	1349
6.2.4.12	Trace	1350
6.2.4.13	X	1351
6.2.4.13.1	Slimits	1352
6.2.4.13.1.1	Left	1353
6.2.4.13.1.2	Right	1354
6.2.4.13.1.3	State	1355
6.2.4.13.1.4	Zoom	1356
6.2.4.13.1.5	State	1356
6.2.4.13.2	Ssize	1357
6.2.4.14	Y	1358
6.2.4.14.1	Percent	1359
6.2.4.15	Z	1360
6.2.5	Math	1360
6.2.5.1	Expression	1360
6.2.5.1.1	Define	1361
6.2.5.2	Mode	1362
6.2.5.3	Position	1362
6.2.5.4	State	1363
6.2.6	Msra	1364
6.2.6.1	Aline	1364
6.2.6.1.1	Show	1364
6.2.6.1.2	Value	1365
6.2.6.2	Window	1366
6.2.6.2.1	Ival	1366
6.2.7	PeakSearch	1367
6.2.7.1	Auto	1367
6.2.7.2	Margin	1367
6.2.7.3	Pshow	1368
6.2.7.4	Subranges	1369
6.2.8	Pmeter<PowerMeter>	1369
6.2.8.1	Relative	1370
6.2.8.1.1	Magnitude	1370
6.2.8.1.2	State	1371
6.2.9	Rtms	1372
6.2.9.1	Aline	1372
6.2.9.1.1	Show	1372
6.2.9.1.2	Value	1373

6.2.9.2	Window	1374
6.2.9.2.1	Ival	1374
6.2.10	Spectrogram	1375
6.2.10.1	Clear	1375
6.2.10.1.1	Immediate	1375
6.2.10.2	Continuous	1376
6.2.10.3	Frame	1376
6.2.10.3.1	Count	1377
6.2.10.3.2	Select	1377
6.2.10.4	Hdepth	1378
6.2.10.5	State	1379
6.2.10.6	ThreeDim	1379
6.2.10.6.1	State	1380
6.2.10.7	Tstamp	1380
6.2.10.7.1	Data	1381
6.2.10.7.2	State	1381
6.2.11	Statistics	1382
6.2.11.1	AmplitudeProbDensity	1383
6.2.11.1.1	State	1383
6.2.11.2	CumulativeDistribFnc	1384
6.2.11.2.1	State	1384
6.2.11.2.2	X<Trace>	1385
6.2.11.3	Nsamples	1386
6.2.11.4	Result<Trace>	1386
6.2.11.5	Scale	1387
6.2.11.5.1	X	1388
6.2.11.5.1.1	Range	1388
6.2.11.5.1.2	RefLevel	1389
6.2.11.5.2	Y	1389
6.2.11.5.2.1	Lower	1390
6.2.11.5.2.2	Unit	1390
6.2.11.5.2.3	Upper	1391
6.2.12	Threshold	1392
6.2.12.1	State	1393
6.2.13	Unit	1393
6.2.13.1	Angle	1394
6.2.13.2	Power	1395
6.3	Calibration	1395
6.3.1	Aiq	1396
6.3.1.1	HaTiming	1396
6.3.1.1.1	State	1396
6.3.2	All	1397
6.3.3	Due	1397
6.3.3.1	Days	1398
6.3.3.2	Schedule	1399
6.3.3.3	Shutdown	1400
6.3.3.4	Time	1400
6.3.3.5	Warmup	1401
6.3.4	PreSelection	1401
6.3.5	Result	1402
6.4	Configure	1402
6.4.1	Ademod	1402
6.4.1.1	Results	1403
6.4.1.1.1	Am	1403

	6.4.1.1.1.1	Detector<Trace>	1403
	6.4.1.1.1.2	Mode	1404
	6.4.1.1.1.3	Reference	1405
	6.4.1.1.1.4	MeastoRef<RefMeasurement>	1406
	6.4.1.1.1.5	State	1407
	6.4.1.1.2	Fm	1407
	6.4.1.1.2.1	Detector<Trace>	1408
	6.4.1.1.2.2	Mode	1408
	6.4.1.1.2.3	Reference	1409
	6.4.1.1.2.4	MeastoRef<RefMeasurement>	1410
	6.4.1.1.2.5	State	1411
	6.4.1.1.3	Pm	1412
	6.4.1.1.3.1	Detector<Trace>	1412
	6.4.1.1.3.2	Mode	1412
	6.4.1.1.3.3	Reference	1413
	6.4.1.1.3.4	MeastoRef<RefMeasurement>	1414
	6.4.1.1.3.5	State	1415
	6.4.1.1.4	Unit	1416
6.4.2	Generator		1417
	6.4.2.1	Connection	1417
	6.4.2.1.1	Cstate	1417
	6.4.2.1.2	State	1418
	6.4.2.2	IpConnection	1418
	6.4.2.2.1	Address	1418
	6.4.2.3	Recording	1419
	6.4.2.3.1	Combine	1419
6.5	Diagnostic		1420
	6.5.1	Hums	1420
	6.5.1.1	All	1421
	6.5.1.2	Bios	1421
	6.5.1.3	Device	1422
	6.5.1.3.1	History	1422
	6.5.1.4	Equipment	1423
	6.5.1.5	FormatPy	1423
	6.5.1.6	Security	1424
	6.5.1.7	Service	1424
	6.5.1.8	State	1424
	6.5.1.9	Storage	1425
	6.5.1.10	Sw	1425
	6.5.1.11	System	1426
	6.5.1.11.1	Info	1426
	6.5.1.11.2	Status	1426
	6.5.1.11.2.1	Summary	1427
	6.5.1.12	Tags	1427
	6.5.1.12.1	All	1428
	6.5.1.12.2	Value	1428
	6.5.1.13	Utilization	1429
	6.5.1.13.1	Activity	1430
	6.5.1.13.1.1	Tracking	1430
	6.5.1.13.1.2	State	1431
	6.5.1.13.2	History	1432
6.5.2	Info		1432
	6.5.2.1	Ccount	1433
6.5.3	Service		1433

6.5.3.1	BiosInfo	1434
6.5.3.2	Calibration	1434
6.5.3.2.1	Date	1434
6.5.3.2.2	Due	1435
6.5.3.2.2.1	Date	1435
6.5.3.2.2.2	State	1436
6.5.3.2.3	Interval	1436
6.5.3.3	Date	1437
6.5.3.4	HwInfo	1437
6.5.3.5	InputPy	1438
6.5.3.5.1	Aiq	1438
6.5.3.5.1.1	TypePy	1438
6.5.3.5.2	Emi	1439
6.5.3.5.2.1	Cfrequency	1439
6.5.3.5.2.2	Spectrum	1440
6.5.3.5.3	Mc	1440
6.5.3.5.3.1	Cfrequency	1441
6.5.3.5.3.2	Distance	1441
6.5.3.5.3.3	Range	1442
6.5.3.5.4	Pulsed	1442
6.5.3.5.4.1	Cfrequency	1443
6.5.3.5.4.2	McFrequency	1443
6.5.3.5.4.3	WbFrequency	1444
6.5.3.5.5	Rf	1444
6.5.3.5.5.1	Cfrequency	1445
6.5.3.5.5.2	Cpath	1445
6.5.3.5.5.3	Spectrum	1446
6.5.3.5.6	Select	1447
6.5.3.5.7	SynthTwo	1447
6.5.3.5.7.1	Frequency	1447
6.5.3.6	Nsource	1448
6.5.3.7	Sfunction	1449
6.5.3.7.1	LastResult	1450
6.5.3.7.2	Results	1450
6.5.3.7.2.1	Save	1451
6.5.3.8	Sinfo	1451
6.5.3.9	SpCheck	1452
6.5.3.9.1	Execute	1452
6.5.3.9.2	Result	1453
6.5.3.10	State	1453
6.5.3.11	Stest	1453
6.5.3.11.1	Result	1454
6.5.3.12	VersionInfo	1454
6.6	Display	1455
6.6.1	Annotation	1455
6.6.1.1	Cbar	1455
6.6.1.2	Frequency	1456
6.6.2	Atab	1456
6.6.3	Blighting	1457
6.6.4	Cmap<Item>	1457
6.6.4.1	Default<Colors>	1458
6.6.4.2	Hsl	1459
6.6.4.3	Pdefined	1460
6.6.5	Faccess	1460

6.6.5.1	Position	1461
6.6.6	FormatPy	1461
6.6.7	Iterm	1462
6.6.7.1	State	1462
6.6.8	Logo	1463
6.6.9	Minfo	1463
6.6.9.1	State	1464
6.6.10	PreSelector	1464
6.6.10.1	Position	1464
6.6.11	Sbar	1465
6.6.11.1	State	1465
6.6.12	Skeys	1466
6.6.12.1	State	1466
6.6.13	Tbar	1467
6.6.13.1	State	1467
6.6.14	Theme	1467
6.6.14.1	Catalog	1468
6.6.14.2	Select	1468
6.6.15	Touchscreen	1469
6.6.15.1	State	1469
6.6.16	Window<Window>	1470
6.6.16.1	Minfo	1470
6.6.16.1.1	State	1470
6.6.16.2	Mtable	1471
6.6.16.3	Size	1472
6.6.16.4	Spectrogram	1473
6.6.16.4.1	Color	1473
6.6.16.4.1.1	Default	1473
6.6.16.4.1.2	Lower	1474
6.6.16.4.1.3	Shape	1474
6.6.16.4.1.4	Style	1475
6.6.16.4.1.5	Upper	1476
6.6.16.5	State	1476
6.6.16.6	Statistics	1477
6.6.16.6.1	CumulativeDistribFnc	1477
6.6.16.6.1.1	Gauss	1478
6.6.16.7	Subwindow<SubWindow>	1479
6.6.16.7.1	Trace<Trace>	1479
6.6.16.7.1.1	Mode	1479
6.6.16.7.1.2	Hcontinuous	1481
6.6.16.7.1.3	Select	1482
6.6.16.7.1.4	State	1483
6.6.16.7.1.5	X	1484
6.6.16.7.1.6	Spacing	1484
6.6.16.7.1.7	Y	1485
6.6.16.7.1.8	Scale	1485
6.6.16.7.1.9	Auto	1487
6.6.16.7.1.10	Mode	1487
6.6.16.7.1.11	Pdivision	1488
6.6.16.7.1.12	RefLevel	1489
6.6.16.7.1.13	Offset	1491
6.6.16.7.1.14	RefPosition	1492
6.6.16.7.1.15	Rvalue	1493
6.6.16.7.1.16	Spacing	1494

	6.6.16.7.2	Zoom	1495
	6.6.16.7.2.1	Area	1495
	6.6.16.7.2.2	Multiple<ZoomWindow>	1497
	6.6.16.7.2.3	Area	1497
	6.6.16.7.2.4	State	1499
	6.6.16.7.2.5	State	1500
	6.6.16.8	Time	1501
	6.6.16.8.1	FormatPy	1502
	6.6.16.9	Trace<Trace>	1503
	6.6.16.9.1	Length	1503
	6.6.16.9.2	Mode	1504
	6.6.16.9.2.1	Hcontinuous	1505
	6.6.16.9.3	Smoothing	1506
	6.6.16.9.3.1	Aperture	1506
	6.6.16.9.3.2	State	1507
	6.6.16.9.4	State	1508
	6.6.16.9.5	Y	1509
	6.6.16.9.5.1	Scale	1509
	6.6.16.9.5.2	Auto	1510
	6.6.16.9.5.3	Maximum	1511
	6.6.16.9.5.4	Minimum	1512
	6.6.16.9.5.5	Mode	1513
	6.6.16.9.5.6	Pdivision	1514
	6.6.16.9.5.7	RefLevel	1515
	6.6.16.9.5.8	Offset	1516
	6.6.16.9.5.9	RefPosition	1517
	6.6.16.9.5.10	Rvalue	1518
	6.6.16.9.5.11	Spacing	1519
	6.6.17	Wselect	1520
6.7	Fetch		1520
	6.7.1	Pmeter<PowerMeter>	1520
6.8	FormatPy		1521
	6.8.1	Data	1521
	6.8.2	Dexport	1522
	6.8.2.1	Dseparator	1522
	6.8.2.2	Header	1523
	6.8.2.3	Traces	1524
	6.8.2.4	Xdistrib	1524
6.9	HardCopy		1525
	6.9.1	Cmap<Item>	1526
	6.9.1.1	Default<Colors>	1526
	6.9.1.2	Hsl	1527
	6.9.1.3	Pdefined	1528
	6.9.2	Comment	1529
	6.9.3	Content	1529
	6.9.4	Copies	1530
	6.9.5	Destination	1531
	6.9.6	Device	1532
	6.9.6.1	Color	1532
	6.9.6.2	Language	1532
	6.9.7	Immediate	1533
	6.9.7.1	Next	1534
	6.9.8	Item	1534
	6.9.8.1	All	1535

6.9.8.2	Window	1535
6.9.8.2.1	Text	1536
6.9.8.2.2	Trace	1536
6.9.8.2.2.1	State	1536
6.9.9	Logo	1537
6.9.10	Mode	1538
6.9.11	Page	1538
6.9.11.1	Count	1538
6.9.11.1.1	State	1539
6.9.11.2	Margin	1539
6.9.11.2.1	Bottom	1540
6.9.11.2.2	Left	1540
6.9.11.2.3	Right	1541
6.9.11.2.4	Top	1542
6.9.11.2.5	Unit	1542
6.9.11.3	Orientation	1543
6.9.11.4	Window	1543
6.9.11.4.1	Channel	1544
6.9.11.4.1.1	State	1544
6.9.11.4.2	Count	1545
6.9.11.4.3	Scale	1545
6.9.11.4.4	State	1546
6.9.12	Print	1547
6.9.13	TdDstamp	1548
6.9.13.1	State	1548
6.9.14	Theme	1549
6.9.14.1	Select	1549
6.9.15	Treport	1549
6.9.15.1	Append	1550
6.9.15.2	Description	1550
6.9.15.3	Item	1551
6.9.15.3.1	Default	1551
6.9.15.3.2	Header	1552
6.9.15.3.2.1	Line<Line>	1552
6.9.15.3.2.2	Control	1552
6.9.15.3.2.3	Text	1553
6.9.15.3.2.4	Title	1554
6.9.15.3.2.5	State	1554
6.9.15.3.3	ListPy	1555
6.9.15.3.4	Logo	1555
6.9.15.3.4.1	Control	1556
6.9.15.3.5	Select	1557
6.9.15.3.6	Template	1557
6.9.15.3.6.1	Catalog	1558
6.9.15.4	New	1558
6.9.15.5	Pagecount	1559
6.9.15.5.1	State	1559
6.9.15.6	PageSize	1560
6.9.15.7	Pcolors	1560
6.9.15.7.1	State	1560
6.9.15.8	TdDstamp	1561
6.9.15.8.1	State	1561
6.9.15.9	Test	1562
6.9.15.9.1	Remove	1562

	6.9.15.9.1.1	All	1563
	6.9.15.9.1.2	Selected	1563
	6.9.15.9.2	Select	1564
	6.9.15.9.2.1	All	1564
	6.9.15.9.2.2	Invert	1565
	6.9.15.9.2.3	NonePy	1565
	6.9.15.10	Title	1566
	6.9.15.10.1	State	1567
6.10	Initiate		1567
6.10.1	Block		1568
6.10.1.1	ConMeas		1569
6.10.1.2	Immediate		1569
6.10.2	ConMeas		1570
6.10.3	Continuous		1570
6.10.4	Espectrum		1571
6.10.5	Sequencer		1571
6.10.5.1	Immediate		1572
6.10.5.2	Mode		1573
6.10.5.3	Refresh		1573
6.10.5.3.1	All		1573
6.10.6	Spurious		1574
6.11	InputPy<InputIx>		1575
6.11.1	Attenuation		1575
6.11.1.1	Auto		1576
6.11.1.2	Protection		1576
6.11.2	Connector		1577
6.11.3	Coupling		1577
6.11.4	Diq		1578
6.11.4.1	Cdevice		1578
6.11.4.2	Range		1578
6.11.4.2.1	Coupling		1579
6.11.4.2.2	Upper		1579
6.11.4.2.2.1	Auto		1580
6.11.4.2.2.2	Unit		1580
6.11.4.3	SymbolRate		1581
6.11.4.3.1	Auto		1582
6.11.5	Dpath		1582
6.11.6	Eatt		1583
6.11.6.1	Auto		1583
6.11.6.2	State		1584
6.11.7	Egain		1584
6.11.7.1	State		1584
6.11.8	File		1585
6.11.8.1	Path		1585
6.11.8.2	Zpadding		1586
6.11.9	FilterPy		1587
6.11.9.1	Hpass		1587
6.11.9.1.1	State		1587
6.11.9.2	Yig		1588
6.11.9.2.1	State		1588
6.11.10	Gain		1589
6.11.10.1	State		1589
6.11.10.2	Value		1590
6.11.11	Impedance		1590

6.11.12	Iq	1591
6.11.12.1	Balanced	1591
6.11.12.1.1	State	1592
6.11.12.2	Fullscale	1592
6.11.12.2.1	Auto	1592
6.11.12.2.2	Level	1593
6.11.12.3	Osc	1593
6.11.12.3.1	Balanced	1594
6.11.12.3.1.1	State	1594
6.11.12.3.2	Fullscale	1595
6.11.12.3.2.1	Level	1595
6.11.12.3.3	Skew	1596
6.11.12.3.3.1	Icomponent	1596
6.11.12.3.3.2	Inverted	1597
6.11.12.3.3.3	Qcomponent	1598
6.11.12.3.3.4	Inverted	1598
6.11.12.3.4	State	1599
6.11.12.3.5	SymbolRate	1600
6.11.12.3.6	Tcpip	1601
6.11.12.3.7	TypePy	1601
6.11.12.4	TypePy	1602
6.11.13	Loscillator	1603
6.11.13.1	Source	1603
6.11.13.1.1	External	1604
6.11.13.1.1.1	Level	1604
6.11.14	Sanalyzer	1605
6.11.14.1	Attenuation	1605
6.11.14.1.1	Auto	1606
6.11.15	Select	1607
6.11.16	Terminator	1607
6.11.17	Uport	1608
6.11.17.1	State	1608
6.11.17.2	Value	1609
6.12	Instrument	1609
6.12.1	Couple	1610
6.12.1.1	AbImpedance	1611
6.12.1.2	AcDc	1611
6.12.1.3	Atten	1612
6.12.1.4	Aunit	1612
6.12.1.5	Bandwidth	1613
6.12.1.6	Center	1613
6.12.1.7	Demod	1614
6.12.1.8	Gain	1614
6.12.1.9	Generator	1615
6.12.1.9.1	Center	1615
6.12.1.9.1.1	Offset	1615
6.12.1.9.1.2	State	1616
6.12.1.9.2	RefLevel	1617
6.12.1.9.2.1	Offset	1617
6.12.1.9.2.2	State	1617
6.12.1.9.3	State	1618
6.12.1.10	Impedance	1619
6.12.1.11	Limit	1619
6.12.1.12	Llines	1620

	6.12.1.13 Marker	1620
	6.12.1.14 Presel	1621
	6.12.1.15 RefLevel	1621
	6.12.1.16 Span	1622
	6.12.1.17 Vbw	1622
6.12.2	Create	1623
	6.12.2.1 Duplicate	1623
	6.12.2.2 New	1623
	6.12.2.3 Replace	1624
6.12.3	ListPy	1624
6.12.4	Mode	1625
6.12.5	Nselect	1625
6.12.6	Rename	1626
6.12.7	Select	1626
6.12.8	SelectName	1627
6.13	Layout	1627
6.13.1	Add	1628
	6.13.1.1 Window	1628
6.13.2	Catalog	1629
	6.13.2.1 Window	1629
6.13.3	Direction	1629
6.13.4	Identify	1630
	6.13.4.1 Window	1630
6.13.5	Remove	1631
	6.13.5.1 Window	1631
6.13.6	Replace	1631
	6.13.6.1 Window	1632
6.13.7	Splitter	1632
6.14	MassMemory	1633
6.14.1	Catalog	1634
	6.14.1.1 Long	1635
6.14.2	Clear	1635
	6.14.2.1 State	1636
6.14.3	Comment	1636
6.14.4	CurrentDirectory	1637
6.14.5	Delete	1637
	6.14.5.1 Immediate	1637
6.14.6	DeleteDirectory	1638
6.14.7	Load<Window>	1638
	6.14.7.1 Auto	1639
	6.14.7.2 Iq	1639
	6.14.7.2.1 State	1639
	6.14.7.2.2 Stream	1640
	6.14.7.2.2.1 Auto	1640
	6.14.7.2.2.2 ListPy	1641
	6.14.7.3 State	1641
	6.14.7.4 Tfactor	1642
	6.14.7.5 TypePy	1642
6.14.8	MakeDirectory	1643
6.14.9	Msis	1643
6.14.10	Name	1644
6.14.11	Network	1645
	6.14.11.1 Disconnect	1645
	6.14.11.2 Map	1646

6.14.11.3 UnusedDrives	1647
6.14.11.4 UsedDrives	1647
6.14.12 Raw	1648
6.14.13 Select	1648
6.14.13.1 Channel	1649
6.14.13.1.1 Item	1649
6.14.13.1.1.1 All	1649
6.14.13.1.1.2 Default	1650
6.14.13.1.1.3 HwSettings	1651
6.14.13.1.1.4 Lines	1652
6.14.13.1.1.5 All	1652
6.14.13.1.1.6 NonePy	1653
6.14.13.1.1.7 ScData	1653
6.14.13.1.1.8 Spectrogram	1654
6.14.13.1.1.9 Trace	1655
6.14.13.1.1.10 Active	1655
6.14.13.1.1.11 Transducer	1656
6.14.13.1.1.12 All	1656
6.14.13.1.1.13 Weighting	1656
6.14.13.2 Item	1657
6.14.13.2.1 All	1657
6.14.13.2.2 Cdata	1658
6.14.13.2.3 Csetup	1659
6.14.13.2.4 Default	1659
6.14.13.2.5 Final	1660
6.14.13.2.6 HardCopy	1660
6.14.13.2.7 HwSettings	1661
6.14.13.2.8 Lines	1662
6.14.13.2.8.1 Active	1662
6.14.13.2.8.2 All	1663
6.14.13.2.9 Macros	1663
6.14.13.2.10NonePy	1664
6.14.13.2.11ScData	1664
6.14.13.2.12Spectrogram	1665
6.14.13.2.13Trace	1666
6.14.13.2.13.1 Active	1666
6.14.13.2.14Transducer	1667
6.14.13.2.14.1 Active	1667
6.14.13.2.14.2 All	1667
6.14.13.2.15ViqData	1668
6.14.13.2.16Weighting	1668
6.14.14 Store<Store>	1669
6.14.14.1 Iq	1669
6.14.14.1.1 Comment	1670
6.14.14.1.2 FormatPy	1671
6.14.14.1.3 Range	1671
6.14.14.1.4 State	1672
6.14.14.2 ListPy	1672
6.14.14.3 Peak	1673
6.14.14.4 Spectrogram	1674
6.14.14.5 Spurious	1674
6.14.14.6 State	1675
6.14.14.6.1 Next	1676
6.14.14.7 Table	1676

6.14.14.7.1	Limit	1677
6.14.14.8	Tfactor	1678
6.14.14.9	Trace	1679
6.14.14.10	TypePy	1679
6.15	Output<OutputConnector>	1680
6.15.1	Ademod	1680
6.15.1.1	Online	1681
6.15.1.1.1	Af	1681
6.15.1.1.1.1	Cfrequency	1681
6.15.1.1.2	Source	1682
6.15.1.1.3	State	1683
6.15.2	Diq	1684
6.15.2.1	State	1684
6.15.3	Ifreq	1684
6.15.3.1	IfFrequency	1685
6.15.3.2	Source	1686
6.15.4	Probe<Probe>	1687
6.15.4.1	Power	1687
6.15.5	Trigger<TriggerPort>	1688
6.15.5.1	Direction	1688
6.15.5.2	Level	1689
6.15.5.3	Otype	1690
6.15.5.4	Pulse	1691
6.15.5.4.1	Immediate	1691
6.15.5.4.2	Length	1691
6.15.6	Uport	1692
6.15.6.1	State	1692
6.15.6.2	Value	1693
6.16	Read	1694
6.16.1	Pmeter<PowerMeter>	1695
6.17	Sense	1695
6.17.1	Ademod	1696
6.17.1.1	AdcPrefilter	1696
6.17.1.2	Af	1697
6.17.1.2.1	Center	1697
6.17.1.2.2	Coupling	1697
6.17.1.2.3	Span	1698
6.17.1.2.3.1	Full	1699
6.17.1.2.4	Start	1699
6.17.1.2.5	Stop	1700
6.17.1.3	Am	1700
6.17.1.3.1	Absolute	1701
6.17.1.3.1.1	AfSpectrum	1701
6.17.1.3.1.2	Result	1701
6.17.1.3.1.3	TypePy	1702
6.17.1.3.1.4	Tdomain	1703
6.17.1.3.1.5	Result	1703
6.17.1.3.1.6	TypePy	1704
6.17.1.3.2	Relative	1704
6.17.1.3.2.1	AfSpectrum	1705
6.17.1.3.2.2	Result	1705
6.17.1.3.2.3	TypePy	1706
6.17.1.3.2.4	Tdomain	1707
6.17.1.3.2.5	Result	1707

6.17.1.3.2.6	TypePy	1708
6.17.1.4	Fm	1708
6.17.1.4.1	AfSpectrum	1709
6.17.1.4.1.1	Result	1709
6.17.1.4.1.2	TypePy	1710
6.17.1.4.2	Offset	1710
6.17.1.4.3	Tdomain	1711
6.17.1.4.3.1	Result	1711
6.17.1.4.3.2	TypePy	1712
6.17.1.5	Mtime	1713
6.17.1.6	Pm	1713
6.17.1.6.1	AfSpectrum	1713
6.17.1.6.1.1	Result	1714
6.17.1.6.1.2	TypePy	1715
6.17.1.6.2	Rpoint	1715
6.17.1.6.2.1	X	1715
6.17.1.6.2.2	Mode	1716
6.17.1.6.3	Tdomain	1717
6.17.1.6.3.1	Result	1717
6.17.1.6.3.2	TypePy	1718
6.17.1.7	Preset	1719
6.17.1.7.1	Restore	1719
6.17.1.7.2	Standard	1719
6.17.1.7.3	Store	1720
6.17.1.8	Set	1721
6.17.1.9	Spectrum	1722
6.17.1.9.1	Bandwidth	1722
6.17.1.9.1.1	Resolution	1722
6.17.1.9.2	Result	1723
6.17.1.9.3	Span	1724
6.17.1.9.3.1	Maximum	1724
6.17.1.9.3.2	Zoom	1725
6.17.1.9.4	TypePy	1725
6.17.1.10	Squelch	1727
6.17.1.10.1	Level	1727
6.17.1.10.2	State	1728
6.17.1.11	Zoom	1728
6.17.1.11.1	Length	1728
6.17.1.11.1.1	Mode	1729
6.17.1.11.2	Start	1730
6.17.1.11.3	State	1731
6.17.2	Adjust	1731
6.17.2.1	All	1732
6.17.2.2	Configure	1733
6.17.2.2.1	Duration	1733
6.17.2.2.1.1	Mode	1734
6.17.2.2.2	Frequency	1734
6.17.2.2.2.1	Limit	1734
6.17.2.2.2.2	High	1735
6.17.2.2.2.3	Low	1735
6.17.2.2.3	Hysteresis	1736
6.17.2.2.3.1	Lower	1736
6.17.2.2.3.2	Upper	1737
6.17.2.2.4	Level	1738

6.17.2.2.4.1	Duration	1738
6.17.2.2.4.2	Mode	1739
6.17.2.2.4.3	Threshold	1740
6.17.2.2.5	Trigger	1740
6.17.2.3	Frequency	1741
6.17.2.4	Level	1742
6.17.2.5	Scale	1742
6.17.2.5.1	Y	1742
6.17.2.5.1.1	Auto	1743
6.17.2.5.1.2	Continuous	1743
6.17.3	Average	1744
6.17.3.1	Count	1744
6.17.3.2	State<Status>	1745
6.17.3.3	TypePy	1746
6.17.4	Bandwidth	1746
6.17.4.1	Demod	1747
6.17.4.1.1	TypePy	1747
6.17.4.2	Resolution	1748
6.17.4.2.1	Auto	1749
6.17.4.2.2	Fft	1749
6.17.4.2.3	Ratio	1750
6.17.4.2.4	TypePy	1750
6.17.4.3	Video	1751
6.17.4.3.1	Auto	1752
6.17.4.3.2	Ratio	1752
6.17.4.3.3	TypePy	1753
6.17.5	Correction	1753
6.17.5.1	Collect	1754
6.17.5.1.1	Acquire	1754
6.17.5.2	Cvl	1755
6.17.5.2.1	Band	1755
6.17.5.2.2	Bias	1756
6.17.5.2.3	Catalog	1757
6.17.5.2.4	Comment	1757
6.17.5.2.5	Data	1758
6.17.5.2.6	Harmonic	1759
6.17.5.2.7	Mixer	1759
6.17.5.2.8	Ports	1760
6.17.5.2.9	Select	1761
6.17.5.2.10	Snumber	1761
6.17.5.3	Fresponse	1762
6.17.5.3.1	Baseband	1762
6.17.5.3.1.1	User	1762
6.17.5.3.1.2	Adjust	1763
6.17.5.3.1.3	RefLevel	1763
6.17.5.3.1.4	State	1764
6.17.5.3.1.5	Flist<FileList>	1764
6.17.5.3.1.6	Catalog	1765
6.17.5.3.1.7	Insert	1766
6.17.5.3.1.8	Magnitude	1766
6.17.5.3.1.9	State	1767
6.17.5.3.1.10	Phase	1767
6.17.5.3.1.11	State	1768
6.17.5.3.1.12	Remove	1769

6.17.5.3.1.13	Select	1769
6.17.5.3.1.14	Size	1770
6.17.5.3.1.15	Refresh	1770
6.17.5.3.1.16	Slist<TouchStone>	1771
6.17.5.3.1.17	Catalog	1772
6.17.5.3.1.18	Insert	1772
6.17.5.3.1.19	Ports	1773
6.17.5.3.1.20	FromPy	1773
6.17.5.3.1.21	To	1774
6.17.5.3.1.22	Remove	1775
6.17.5.3.1.23	Select	1775
6.17.5.3.1.24	Size	1776
6.17.5.3.1.25	State	1776
6.17.5.3.1.26	State	1777
6.17.5.3.1.27	Store	1777
6.17.5.3.2	InputPy<InputIx>	1778
6.17.5.3.2.1	User	1778
6.17.5.3.2.2	Adjust	1779
6.17.5.3.2.3	RefLevel	1779
6.17.5.3.2.4	State	1779
6.17.5.3.2.5	Flist<FileList>	1780
6.17.5.3.2.6	Catalog	1781
6.17.5.3.2.7	Insert	1782
6.17.5.3.2.8	Magnitude	1783
6.17.5.3.2.9	State	1783
6.17.5.3.2.10	Phase	1784
6.17.5.3.2.11	State	1784
6.17.5.3.2.12	Remove	1785
6.17.5.3.2.13	Select	1785
6.17.5.3.2.14	Size	1786
6.17.5.3.2.15	Refresh	1786
6.17.5.3.2.16	Slist<TouchStone>	1787
6.17.5.3.2.17	Catalog	1788
6.17.5.3.2.18	Insert	1789
6.17.5.3.2.19	Ports	1790
6.17.5.3.2.20	FromPy	1790
6.17.5.3.2.21	To	1791
6.17.5.3.2.22	Remove	1792
6.17.5.3.2.23	Select	1792
6.17.5.3.2.24	Size	1793
6.17.5.3.2.25	State	1793
6.17.5.3.2.26	State	1794
6.17.5.3.2.27	Store	1795
6.17.5.3.3	Lsources	1795
6.17.5.3.3.1	State	1795
6.17.5.3.4	User	1796
6.17.5.3.4.1	Adjust	1797
6.17.5.3.4.2	RefLevel	1797
6.17.5.3.4.3	State	1797
6.17.5.3.4.4	Flist<FileList>	1798
6.17.5.3.4.5	Catalog	1799
6.17.5.3.4.6	Data	1799
6.17.5.3.4.7	Frequency	1800
6.17.5.3.4.8	Magnitude	1800

6.17.5.3.4.9	Phase	1801
6.17.5.3.4.10	Insert	1801
6.17.5.3.4.11	Magnitude	1802
6.17.5.3.4.12	State	1802
6.17.5.3.4.13	Phase	1803
6.17.5.3.4.14	State	1803
6.17.5.3.4.15	Remove	1804
6.17.5.3.4.16	Select	1804
6.17.5.3.4.17	Size	1805
6.17.5.3.4.18	Fstate	1805
6.17.5.3.4.19	Iq	1806
6.17.5.3.4.20	Data	1806
6.17.5.3.4.21	Frequency	1806
6.17.5.3.4.22	Magnitude	1807
6.17.5.3.4.23	Phase	1807
6.17.5.3.4.24	Pstate	1807
6.17.5.3.4.25	Scope	1808
6.17.5.3.4.26	Scovered	1808
6.17.5.3.4.27	Slist<TouchStone>	1809
6.17.5.3.4.28	Catalog	1810
6.17.5.3.4.29	Data	1810
6.17.5.3.4.30	Frequency	1811
6.17.5.3.4.31	Magnitude	1811
6.17.5.3.4.32	Phase<SPortPair>	1812
6.17.5.3.4.33	Insert	1812
6.17.5.3.4.34	Ports	1813
6.17.5.3.4.35	FromPy	1813
6.17.5.3.4.36	To	1814
6.17.5.3.4.37	Remove	1815
6.17.5.3.4.38	Select	1815
6.17.5.3.4.39	Size	1816
6.17.5.3.4.40	State	1816
6.17.5.3.4.41	Spectrum	1817
6.17.5.3.4.42	Data	1817
6.17.5.3.4.43	Frequency	1817
6.17.5.3.4.44	Magnitude	1818
6.17.5.3.4.45	Phase	1818
6.17.5.3.4.46	State	1819
6.17.5.3.4.47	Store	1819
6.17.5.3.4.48	Valid	1820
6.17.5.4	Method	1820
6.17.5.5	State	1821
6.17.5.6	Transducer	1821
6.17.5.6.1	Active	1822
6.17.5.6.2	Adjust	1822
6.17.5.6.2.1	RefLevel	1822
6.17.5.6.2.2	State	1823
6.17.5.6.3	Catalog	1823
6.17.5.6.4	Comment	1824
6.17.5.6.5	Data	1824
6.17.5.6.6	Generate	1825
6.17.5.6.7	InputPy<InputIx>	1826
6.17.5.6.7.1	Active	1826
6.17.5.6.7.2	State	1827

6.17.5.6.8	Scaling	1827
6.17.5.6.9	Select	1828
6.17.5.6.10	State	1828
6.17.5.6.11	Unit	1829
6.17.6	Ddemod	1829
6.17.6.1	Search	1830
6.17.6.1.1	Sync	1830
6.17.6.1.1.1	State	1830
6.17.7	Espectrum<SubBlock>	1831
6.17.7.1	Bwid	1831
6.17.7.2	FilterPy	1832
6.17.7.2.1	Rrc	1832
6.17.7.2.1.1	Alpha	1832
6.17.7.2.1.2	State	1833
6.17.7.3	Hspeed	1834
6.17.7.4	Msr	1835
6.17.7.4.1	Apply	1835
6.17.7.4.2	Band	1835
6.17.7.4.3	Bcategory	1836
6.17.7.4.4	ClassPy	1837
6.17.7.4.5	Gsm	1837
6.17.7.4.5.1	Carrier	1838
6.17.7.4.5.2	Cpresent	1839
6.17.7.4.6	Lte	1839
6.17.7.4.6.1	Cpresent	1840
6.17.7.4.7	Mpower	1840
6.17.7.4.8	RfbWidth	1841
6.17.7.5	Preset	1842
6.17.7.5.1	Restore	1842
6.17.7.5.2	Standard	1842
6.17.7.5.3	Store	1843
6.17.7.6	Range<RangePy>	1844
6.17.7.6.1	Bandwidth	1845
6.17.7.6.1.1	Resolution	1845
6.17.7.6.1.2	Video	1846
6.17.7.6.2	Count	1847
6.17.7.6.3	FilterPy	1847
6.17.7.6.3.1	TypePy	1848
6.17.7.6.4	Frequency	1849
6.17.7.6.4.1	Start	1849
6.17.7.6.4.2	Stop	1850
6.17.7.6.5	InputPy	1851
6.17.7.6.5.1	Attenuation	1851
6.17.7.6.5.2	Auto	1852
6.17.7.6.5.3	Gain	1852
6.17.7.6.5.4	State	1853
6.17.7.6.5.5	Value	1854
6.17.7.6.6	Insert	1855
6.17.7.6.7	Limit<LimitIx>	1856
6.17.7.6.7.1	Absolute	1856
6.17.7.6.7.2	Start	1856
6.17.7.6.7.3	Stop	1857
6.17.7.6.7.4	Relative	1858
6.17.7.6.7.5	Start	1859

6.17.7.6.7.6	Abs	1860
6.17.7.6.7.7	Function	1861
6.17.7.6.7.8	Stop	1862
6.17.7.6.7.9	Abs	1863
6.17.7.6.7.10	Function	1864
6.17.7.6.7.11	State	1865
6.17.7.6.8	MI Calc	1866
6.17.7.6.9	RefLevel	1867
6.17.7.6.10	Sweep	1868
6.17.7.6.10.1	Time	1868
6.17.7.6.10.2	Auto	1869
6.17.7.6.11	Transducer	1869
6.17.7.7	Rrange	1870
6.17.7.8	Rtype	1871
6.17.7.9	Scenter	1871
6.17.7.10	Scount	1872
6.17.7.11	Ssetup	1873
6.17.8	FilterPy<FilterPy>	1874
6.17.8.1	Aoff	1874
6.17.8.2	Aweighted	1874
6.17.8.2.1	State	1875
6.17.8.3	Ccir	1876
6.17.8.3.1	Unweighted	1876
6.17.8.3.1.1	State	1876
6.17.8.3.2	Weighted	1877
6.17.8.3.2.1	State	1877
6.17.8.4	Demphasis	1878
6.17.8.4.1	State	1878
6.17.8.4.2	Tconstant	1879
6.17.8.5	Hpass	1880
6.17.8.5.1	Frequency	1880
6.17.8.5.1.1	Absolute	1880
6.17.8.5.1.2	Manual	1881
6.17.8.5.2	State	1882
6.17.8.6	Lpass	1883
6.17.8.6.1	Frequency	1883
6.17.8.6.1.1	Absolute	1883
6.17.8.6.1.2	Manual	1884
6.17.8.6.1.3	Relative	1885
6.17.8.6.2	State	1886
6.17.9	Frequency	1887
6.17.9.1	Annotation	1887
6.17.9.2	Center	1888
6.17.9.2.1	Step	1888
6.17.9.2.1.1	Auto	1889
6.17.9.2.1.2	Link	1889
6.17.9.2.1.3	Factor	1890
6.17.9.3	Offset	1891
6.17.9.4	Span	1892
6.17.9.4.1	Full	1892
6.17.9.5	Start	1893
6.17.9.6	Stop	1894
6.17.10	Iq	1894
6.17.10.1	Bandwidth	1895

6.17.10.1.1	Mode	1895
6.17.10.1.2	Resolution	1896
6.17.11	ListPy	1896
6.17.11.1	Power	1897
6.17.11.1.1	Result	1897
6.17.11.1.2	Sequence	1897
6.17.11.1.3	Set	1898
6.17.11.1.4	State	1899
6.17.11.2	Range<RangePy>	1900
6.17.11.2.1	Bandwidth	1901
6.17.11.2.1.1	Resolution	1901
6.17.11.2.1.2	Video	1902
6.17.11.2.2	BreakPy	1902
6.17.11.2.3	Count	1903
6.17.11.2.4	Detector	1904
6.17.11.2.5	FilterPy	1904
6.17.11.2.5.1	TypePy	1905
6.17.11.2.6	Frequency	1905
6.17.11.2.6.1	Start	1906
6.17.11.2.6.2	Stop	1907
6.17.11.2.7	InputPy	1907
6.17.11.2.7.1	Attenuation	1908
6.17.11.2.7.2	Auto	1909
6.17.11.2.7.3	Gain	1909
6.17.11.2.7.4	State	1909
6.17.11.2.7.5	Value	1910
6.17.11.2.8	Limit	1911
6.17.11.2.8.1	Start	1911
6.17.11.2.8.2	State	1912
6.17.11.2.8.3	Stop	1913
6.17.11.2.9	Points	1913
6.17.11.2.9.1	Value	1914
6.17.11.2.10	RefLevel	1914
6.17.11.2.11	Sweep	1915
6.17.11.2.11.1	Time	1915
6.17.11.2.11.2	Auto	1916
6.17.11.2.12	Transducer	1917
6.17.11.3	Xadjust	1917
6.17.12	Mixer	1918
6.17.12.1	Bias	1918
6.17.12.1.1	High	1918
6.17.12.1.2	Low	1919
6.17.12.2	Frequency	1920
6.17.12.2.1	Handover	1920
6.17.12.2.2	Start	1921
6.17.12.2.3	Stop	1921
6.17.12.3	Harmonic	1922
6.17.12.3.1	Band	1922
6.17.12.3.2	High	1923
6.17.12.3.2.1	State	1924
6.17.12.3.3	Low	1924
6.17.12.3.4	TypePy	1925
6.17.12.4	Ifreq	1926
6.17.12.5	LoPower	1926

6.17.12.6	Loss	1927
6.17.12.6.1	High	1927
6.17.12.6.2	Low	1927
6.17.12.6.3	Table	1928
6.17.12.6.3.1	High	1928
6.17.12.6.3.2	Low	1929
6.17.12.7	Ports	1930
6.17.12.8	RfOverrange	1930
6.17.12.8.1	State	1930
6.17.12.9	Signal	1931
6.17.12.10	State	1932
6.17.12.11	Threshold	1932
6.17.13	Mpower	1933
6.17.13.1	Ftype	1933
6.17.13.2	Result	1934
6.17.13.2.1	ListPy	1934
6.17.13.2.2	Min	1934
6.17.13.3	Sequence	1935
6.17.14	Msra	1936
6.17.14.1	Capture	1936
6.17.14.1.1	Offset	1936
6.17.15	Pmeter<PowerMeter>	1937
6.17.15.1	Dcycle	1937
6.17.15.1.1	State	1938
6.17.15.1.2	Value	1938
6.17.15.2	Frequency	1939
6.17.15.2.1	Link	1940
6.17.15.3	Mtime	1941
6.17.15.3.1	Average	1941
6.17.15.3.1.1	Count	1942
6.17.15.3.1.2	State	1943
6.17.15.4	Roffset	1943
6.17.15.4.1	State	1944
6.17.15.5	Soffset	1944
6.17.15.6	State	1945
6.17.15.7	Trigger	1946
6.17.15.7.1	Dtime	1946
6.17.15.7.2	Holdoff	1947
6.17.15.7.3	Hysteresis	1947
6.17.15.7.4	Level	1948
6.17.15.7.5	Slope	1949
6.17.15.7.6	State	1950
6.17.15.8	Update	1950
6.17.15.8.1	State	1951
6.17.16	Power	1951
6.17.16.1	Achannel	1952
6.17.16.1.1	AcPairs	1952
6.17.16.1.2	AgChannels	1953
6.17.16.1.3	Bandwidth	1953
6.17.16.1.3.1	Gap<GapChannel>	1954
6.17.16.1.3.2	Auto	1954
6.17.16.1.3.3	Manual	1955
6.17.16.1.3.4	Lower	1955
6.17.16.1.3.5	Upper	1956

6.17.16.1.3.6	UaChannel	1957
6.17.16.1.3.7	Ualternate<UpperAltChannel>	1957
6.17.16.1.4	FilterPy	1958
6.17.16.1.4.1	Alpha	1958
6.17.16.1.4.2	Achannel	1959
6.17.16.1.4.3	All	1959
6.17.16.1.4.4	Alternate<Channel>	1960
6.17.16.1.4.5	Channel<Channel>	1961
6.17.16.1.4.6	Gap<GapChannel>	1962
6.17.16.1.4.7	Auto	1962
6.17.16.1.4.8	Manual	1963
6.17.16.1.4.9	Lower	1963
6.17.16.1.4.10	Upper	1964
6.17.16.1.4.11	Sblock<SubBlock>	1965
6.17.16.1.4.12	Channel<Channel>	1965
6.17.16.1.4.13	UaChannel	1966
6.17.16.1.4.14	Ualternate<UpperAltChannel>	1967
6.17.16.1.4.15	State	1968
6.17.16.1.4.16	Achannel	1968
6.17.16.1.4.17	All	1969
6.17.16.1.4.18	Alternate<Channel>	1969
6.17.16.1.4.19	Channel<Channel>	1970
6.17.16.1.4.20	Gap<GapChannel>	1971
6.17.16.1.4.21	Auto	1971
6.17.16.1.4.22	Manual	1972
6.17.16.1.4.23	Lower	1972
6.17.16.1.4.24	Upper	1973
6.17.16.1.4.25	Sblock<SubBlock>	1974
6.17.16.1.4.26	Channel<Channel>	1974
6.17.16.1.4.27	UaChannel	1975
6.17.16.1.4.28	Ualternate<UpperAltChannel>	1976
6.17.16.1.5	Gap<GapChannel>	1977
6.17.16.1.5.1	Auto	1977
6.17.16.1.5.2	Msize	1978
6.17.16.1.5.3	Manual	1978
6.17.16.1.5.4	Channel	1979
6.17.16.1.5.5	Count	1979
6.17.16.1.5.6	Lower	1979
6.17.16.1.5.7	Upper	1980
6.17.16.1.5.8	Mode	1981
6.17.16.1.6	Gchannel	1982
6.17.16.1.6.1	State	1982
6.17.16.1.6.2	Gap<GapChannel>	1982
6.17.16.1.6.3	Manual	1983
6.17.16.1.6.4	Lower	1983
6.17.16.1.6.5	Upper	1984
6.17.16.1.7	Mode	1985
6.17.16.1.8	Name	1985
6.17.16.1.8.1	Achannel	1985
6.17.16.1.8.2	Alternate<Channel>	1986
6.17.16.1.8.3	Channel<Channel>	1987
6.17.16.1.8.4	Gap<GapChannel>	1988
6.17.16.1.8.5	UaChannel	1989
6.17.16.1.8.6	Ualternate<UpperAltChannel>	1990

6.17.16.1.9 PresetRefLevel	1991
6.17.16.1.10Reference	1991
6.17.16.1.10.1 TxChannel	1991
6.17.16.1.10.2 Auto	1992
6.17.16.1.10.3 Manual	1992
6.17.16.1.11Sbcount	1993
6.17.16.1.12Sblock<SubBlock>	1993
6.17.16.1.12.1 Bandwidth	1994
6.17.16.1.12.2 Channel<Channel>	1994
6.17.16.1.12.3 Center	1995
6.17.16.1.12.4 Channel<Channel>	1995
6.17.16.1.12.5 Frequency	1997
6.17.16.1.12.6 Center	1997
6.17.16.1.12.7 Name	1998
6.17.16.1.12.8 Channel<Channel>	1998
6.17.16.1.12.9 RfbWidth	1999
6.17.16.1.12.10Technology	2000
6.17.16.1.12.11Channel<Channel>	2000
6.17.16.1.12.12TxChannel	2001
6.17.16.1.12.13Count	2001
6.17.16.1.13Spacing	2002
6.17.16.1.13.1 Achannel	2002
6.17.16.1.13.2 Alternate<Channel>	2003
6.17.16.1.13.3 Channel<Channel>	2004
6.17.16.1.13.4 Gap<GapChannel>	2005
6.17.16.1.13.5 Auto	2005
6.17.16.1.13.6 Manual	2006
6.17.16.1.13.7 Lower	2006
6.17.16.1.13.8 Upper	2007
6.17.16.1.13.9 UaChannel	2008
6.17.16.1.13.10Ualternate<UpperAltChannel>	2008
6.17.16.1.14Ssetup	2009
6.17.16.1.15TxChannel	2010
6.17.16.1.15.1 Count	2010
6.17.16.2 Bandwidth	2011
6.17.16.3 Hspeed	2011
6.17.16.4 Ncorrection	2012
6.17.16.5 Trace	2013
6.17.17 Probe<Probe>	2013
6.17.17.1 Id	2014
6.17.17.1.1 PartNumber	2014
6.17.17.1.2 SrNumber	2014
6.17.17.2 Setup	2015
6.17.17.2.1 AttRatio	2015
6.17.17.2.2 CmOffset	2016
6.17.17.2.3 DmOffset	2017
6.17.17.2.4 Mode	2017
6.17.17.2.5 Name	2018
6.17.17.2.6 NmOffset	2018
6.17.17.2.7 Pmode	2019
6.17.17.2.8 PmOffset	2020
6.17.17.2.9 State	2021
6.17.17.2.10TypePy	2021
6.17.18 Rlength	2022

6.17.19	Roscillator	2022
6.17.19.1	Coupling	2022
6.17.19.1.1	Bandwidth	2023
6.17.19.1.1.1	Mode	2024
6.17.19.1.2	Mode	2024
6.17.19.2	LbWidth	2025
6.17.19.3	O100	2026
6.17.19.4	O640	2026
6.17.19.5	Osync	2027
6.17.19.6	Output<OutputConnector>	2028
6.17.19.7	PassThrough	2029
6.17.19.8	Source	2029
6.17.19.8.1	Eauto	2030
6.17.19.9	Trange	2031
6.17.20	Rtms	2032
6.17.20.1	Capture	2032
6.17.20.1.1	Offset	2032
6.17.21	Sampling	2033
6.17.21.1	Clkio	2033
6.17.21.1.1	Output	2033
6.17.22	SwapIq	2034
6.17.23	Sweep	2035
6.17.23.1	Count	2035
6.17.23.1.1	Current	2036
6.17.23.2	Duration	2036
6.17.23.3	Egate	2037
6.17.23.3.1	Auto	2037
6.17.23.3.2	Holdoff	2038
6.17.23.3.3	Length	2038
6.17.23.3.4	Level	2039
6.17.23.3.4.1	External<ExternalPort>	2039
6.17.23.3.4.2	IfPower	2040
6.17.23.3.4.3	RfPower	2041
6.17.23.3.5	Polarity	2042
6.17.23.3.6	Source	2042
6.17.23.3.7	Trace<Trace>	2043
6.17.23.3.7.1	Comment	2043
6.17.23.3.7.2	Period	2044
6.17.23.3.7.3	Start	2045
6.17.23.3.7.4	State<Status>	2045
6.17.23.3.7.5	Stop<GateRange>	2047
6.17.23.3.8	TypePy	2048
6.17.23.4	Mode	2049
6.17.23.5	Optimize	2050
6.17.23.6	Points	2050
6.17.23.7	Scapture	2051
6.17.23.7.1	Events	2051
6.17.23.7.2	Gap	2051
6.17.23.7.3	Length	2052
6.17.23.7.3.1	Time	2052
6.17.23.7.4	Offset	2053
6.17.23.7.4.1	Time	2053
6.17.23.7.5	State	2054
6.17.23.8	Time	2054

6.17.23.8.1	Auto	2055
6.17.23.9	TypePy	2055
6.17.23.9.1	Used	2056
6.17.23.10	Window<Window>	2056
6.17.23.10.1	Points	2057
6.17.24	SymbolRate	2058
6.17.25	Trace	2058
6.17.25.1	Iq	2058
6.17.25.1.1	Sync	2059
6.17.25.1.1.1	Mode	2059
6.17.26	Window<Window>	2060
6.17.26.1	Detector<Trace>	2060
6.17.26.1.1	Function	2060
6.17.26.1.1.1	Auto	2061
6.18	Source	2062
6.18.1	Current	2062
6.18.1.1	Auxiliary	2063
6.18.1.1.1	Limit	2063
6.18.1.1.1.1	High	2063
6.18.1.2	Control<Source>	2064
6.18.1.2.1	Limit	2064
6.18.1.2.1.1	High	2064
6.18.1.3	Power<Source>	2065
6.18.1.3.1	Limit	2066
6.18.1.3.1.1	High	2066
6.18.1.4	Sequence	2067
6.18.1.4.1	Result	2067
6.18.2	External	2068
6.18.2.1	Frequency	2068
6.18.2.1.1	Coupling	2069
6.18.2.1.1.1	State	2069
6.18.2.1.2	Factor	2070
6.18.2.1.2.1	Denominator	2070
6.18.2.1.2.2	Numerator	2071
6.18.2.1.3	Offset	2071
6.18.2.1.4	Sweep	2072
6.18.2.1.4.1	State	2072
6.18.2.2	Power	2073
6.18.2.2.1	Level	2073
6.18.2.3	Roscillator	2074
6.18.2.3.1	External	2074
6.18.2.3.1.1	Frequency	2074
6.18.2.3.1.2	Mode	2075
6.18.2.3.2	Source	2076
6.18.2.4	State	2077
6.18.3	Generator	2077
6.18.3.1	Channel	2078
6.18.3.1.1	Coupling	2078
6.18.3.2	DutBypass	2079
6.18.3.3	Frequency	2079
6.18.3.3.1	Step	2080
6.18.3.4	Level	2081
6.18.3.5	Modulation	2081
6.18.3.6	Pulse	2082

6.18.3.6.1	Period	2082
6.18.3.6.2	Trigger	2083
6.18.3.6.2.1	Output	2083
6.18.3.6.3	Width	2084
6.18.3.7	State	2085
6.18.4	Power	2085
6.18.4.1	Level	2086
6.18.4.1.1	Immediate	2086
6.18.4.1.1.1	Offset	2086
6.18.4.2	Sequence	2087
6.18.4.2.1	Result	2087
6.18.5	Temperature	2088
6.18.5.1	Frontend	2088
6.18.6	Voltage	2088
6.18.6.1	Auxiliary	2089
6.18.6.1.1	Level	2089
6.18.6.1.1.1	Amplitude	2089
6.18.6.1.1.2	Limit	2090
6.18.6.1.1.3	High	2090
6.18.6.1.1.4	Low	2091
6.18.6.1.1.5	State	2091
6.18.6.2	Channel	2092
6.18.6.2.1	Coupling	2092
6.18.6.3	Control<Source>	2093
6.18.6.3.1	Level	2093
6.18.6.3.1.1	Amplitude	2093
6.18.6.3.1.2	Limit	2094
6.18.6.3.1.3	High	2094
6.18.6.3.1.4	Low	2095
6.18.6.3.1.5	State	2096
6.18.6.4	Power<Source>	2097
6.18.6.4.1	Level	2097
6.18.6.4.1.1	Amplitude	2097
6.18.6.4.1.2	Limit	2098
6.18.6.4.1.3	High	2098
6.18.6.4.1.4	Low	2099
6.18.6.4.1.5	Mode	2100
6.18.6.4.1.6	State	2101
6.18.6.4.2	Limit	2102
6.18.6.4.2.1	High	2102
6.18.6.5	Sequence	2103
6.18.6.5.1	Result	2103
6.18.6.6	State	2104
6.19	Status	2105
6.19.1	Operation	2105
6.19.1.1	Condition	2106
6.19.1.2	Enable	2106
6.19.1.3	Event	2107
6.19.1.4	Ntransition	2107
6.19.1.5	Pcalibration	2108
6.19.1.5.1	Condition	2108
6.19.1.5.2	Enable	2108
6.19.1.5.3	Event	2109
6.19.1.5.4	Ntransistion	2110

6.19.1.5.5	Ptransistion	2110
6.19.1.6	Ptransition	2111
6.19.2	Questionable	2112
6.19.2.1	AcpLimit	2112
6.19.2.1.1	Condition	2112
6.19.2.1.2	Enable	2113
6.19.2.1.3	Event	2114
6.19.2.1.4	Ntransition	2114
6.19.2.1.5	Ptransition	2115
6.19.2.2	Calibration	2116
6.19.2.2.1	Condition	2116
6.19.2.2.2	Enable	2116
6.19.2.2.3	Event	2117
6.19.2.2.4	Ntransition	2117
6.19.2.2.5	Ptransition	2118
6.19.2.3	Condition	2119
6.19.2.4	Correction	2119
6.19.2.4.1	Condition	2119
6.19.2.4.2	Enable	2120
6.19.2.4.3	Event	2121
6.19.2.4.4	Ntransition	2121
6.19.2.4.5	Ptransition	2122
6.19.2.5	Diq	2122
6.19.2.5.1	Condition	2123
6.19.2.5.2	Enable	2123
6.19.2.5.3	Event	2124
6.19.2.5.4	Ntransition	2124
6.19.2.5.5	Ptransition	2125
6.19.2.6	Enable	2126
6.19.2.7	Event	2126
6.19.2.8	Extended	2127
6.19.2.8.1	Condition	2127
6.19.2.8.2	Enable	2127
6.19.2.8.3	Event	2128
6.19.2.8.4	Info	2129
6.19.2.8.4.1	Condition	2129
6.19.2.8.4.2	Enable	2129
6.19.2.8.4.3	Event	2130
6.19.2.8.4.4	Ntransition	2131
6.19.2.8.4.5	Ptransition	2132
6.19.2.8.5	Ntransition	2133
6.19.2.8.6	Ptransition	2133
6.19.2.9	Frequency	2134
6.19.2.9.1	Condition	2135
6.19.2.9.2	Enable	2135
6.19.2.9.3	Event	2136
6.19.2.9.4	Ntransition	2136
6.19.2.9.5	Ptransition	2137
6.19.2.10	Integrity	2138
6.19.2.10.1	Condition	2138
6.19.2.10.2	Enable	2139
6.19.2.10.3	Event	2139
6.19.2.10.4	Ntransition	2140
6.19.2.10.5	Ptransition	2141

6.19.2.10.6	Signal	2141
6.19.2.10.6.1	Condition	2142
6.19.2.10.6.2	Enable	2142
6.19.2.10.6.3	Event	2143
6.19.2.10.6.4	Ntransition	2143
6.19.2.10.6.5	Ptransition	2144
6.19.2.10.7	Uncalibrated	2145
6.19.2.10.7.1	Condition	2145
6.19.2.10.7.2	Enable	2145
6.19.2.10.7.3	Event	2146
6.19.2.10.7.4	Ntransition	2146
6.19.2.10.7.5	Ptransition	2147
6.19.2.11	Limit<Window>	2148
6.19.2.11.1	Condition	2148
6.19.2.11.2	Enable	2149
6.19.2.11.3	Event	2150
6.19.2.11.4	Ntransition	2150
6.19.2.11.5	Ptransition	2151
6.19.2.12	Lmargin<Window>	2152
6.19.2.12.1	Condition	2153
6.19.2.12.2	Enable	2153
6.19.2.12.3	Event	2154
6.19.2.12.4	Ntransition	2155
6.19.2.12.5	Ptransition	2156
6.19.2.13	Ntransition	2157
6.19.2.14	Pnoise	2157
6.19.2.14.1	Condition	2158
6.19.2.14.2	Enable	2158
6.19.2.14.3	Event	2159
6.19.2.14.4	Ntransition	2159
6.19.2.14.5	Ptransition	2160
6.19.2.15	Power	2161
6.19.2.15.1	Condition	2161
6.19.2.15.2	DcpNoise	2162
6.19.2.15.2.1	Condition	2162
6.19.2.15.2.2	Enable	2162
6.19.2.15.2.3	Event	2163
6.19.2.15.2.4	Ntransition	2164
6.19.2.15.2.5	Ptransition	2165
6.19.2.15.3	Enable	2166
6.19.2.15.4	Event	2167
6.19.2.15.5	Ntransition	2167
6.19.2.15.6	Ptransition	2168
6.19.2.16	Ptransition	2169
6.19.2.17	Sync	2169
6.19.2.17.1	Condition	2170
6.19.2.17.2	Enable	2170
6.19.2.17.3	Event	2171
6.19.2.17.4	Ntransition	2171
6.19.2.17.5	Ptransition	2172
6.19.2.18	Temperature	2173
6.19.2.18.1	Condition	2173
6.19.2.18.2	Enable	2173
6.19.2.18.3	Event	2174

	6.19.2.18.4 Ntransition	2175
	6.19.2.18.5 Ptransition	2176
	6.19.2.19 Time	2177
	6.19.2.19.1 Condition	2177
	6.19.2.19.2 Enable	2177
	6.19.2.19.3 Event	2178
	6.19.2.19.4 Ntransition	2179
	6.19.2.19.5 Ptransition	2179
	6.19.2.20 Transducer	2180
	6.19.2.20.1 Condition	2181
	6.19.2.20.2 Enable	2181
	6.19.2.20.3 Event	2182
	6.19.2.20.4 Ntransition	2182
	6.19.2.20.5 Ptransition	2183
	6.19.3 Queue	2184
	6.19.3.1 Next	2184
6.20	System	2185
6.20.1	Clogging	2185
6.20.2	Communicate	2186
6.20.2.1	Gpib	2186
6.20.2.1.1	Rdevice	2186
6.20.2.1.1.1	Generator<Generator>	2186
6.20.2.1.1.2	Address	2187
6.20.2.1.2	Self	2188
6.20.2.1.2.1	Address	2188
6.20.2.1.2.2	Rterminator	2188
6.20.2.2	Internal	2189
6.20.2.2.1	Command	2189
6.20.2.2.1.1	Tables	2190
6.20.2.2.2	Completed	2190
6.20.2.2.2.1	Event	2191
6.20.2.3	Rdevice	2191
6.20.2.3.1	Generator<Generator>	2191
6.20.2.3.1.1	Interface	2192
6.20.2.3.1.2	Link	2193
6.20.2.3.1.3	TypePy	2193
6.20.2.3.2	Oscilloscope	2194
6.20.2.3.2.1	Alignment	2194
6.20.2.3.2.2	Date	2195
6.20.2.3.2.3	Step<Step>	2195
6.20.2.3.2.4	State	2195
6.20.2.3.2.5	Idn	2196
6.20.2.3.2.6	LedState	2196
6.20.2.3.2.7	State	2197
6.20.2.3.2.8	Tcpip	2197
6.20.2.3.2.9	Vdevice	2198
6.20.2.3.2.10	Vfirmware	2198
6.20.2.3.3	Pmeter<PowerMeter>	2199
6.20.2.3.3.1	Configure	2199
6.20.2.3.3.2	Auto	2199
6.20.2.3.3.3	State	2200
6.20.2.3.3.4	Count	2201
6.20.2.3.3.5	Define	2201
6.20.2.4	Rest	2202

6.20.2.4.1	Enable	2202
6.20.2.5	Snmp	2203
6.20.2.5.1	Community	2203
6.20.2.5.1.1	Ro	2203
6.20.2.5.1.2	Rw	2204
6.20.2.5.2	Contact	2204
6.20.2.5.3	Location	2205
6.20.2.5.4	Usm	2206
6.20.2.5.4.1	User	2206
6.20.2.5.4.2	All	2207
6.20.2.5.5	Version	2208
6.20.2.6	Tcpip	2208
6.20.2.6.1	Rdevice	2209
6.20.2.6.1.1	Generator<Generator>	2209
6.20.2.6.1.2	Address	2209
6.20.3	Compatible	2210
6.20.4	DeviceFootprint	2211
6.20.5	Display	2211
6.20.5.1	Fpanel	2211
6.20.5.1.1	State	2212
6.20.5.2	Lock	2212
6.20.5.3	Message	2213
6.20.5.3.1	State	2213
6.20.5.3.2	Text	2214
6.20.5.4	Update	2214
6.20.6	Error	2215
6.20.6.1	All	2216
6.20.6.2	Clear	2216
6.20.6.2.1	Remote	2216
6.20.6.3	Display	2217
6.20.6.4	ListPy	2218
6.20.7	Firmware	2218
6.20.7.1	Update	2219
6.20.8	FormatPy	2219
6.20.8.1	Ident	2220
6.20.9	Help	2220
6.20.9.1	Headers	2221
6.20.9.2	Syntax	2221
6.20.9.2.1	All	2222
6.20.10	Identify	2222
6.20.10.1	Factory	2222
6.20.10.2	String	2223
6.20.11	IfGain	2223
6.20.11.1	Mode	2224
6.20.12	Language	2224
6.20.13	Lxi	2225
6.20.13.1	Info	2225
6.20.13.2	LanReset	2226
6.20.13.3	Mdescription	2226
6.20.13.4	Password	2227
6.20.14	Option	2227
6.20.14.1	License	2227
6.20.14.1.1	ListPy	2228
6.20.14.2	Trial	2228

6.20.14.2.1	ListPy	2229
6.20.14.2.2	State	2229
6.20.15	Osystem	2230
6.20.16	Plugin	2230
6.20.16.1	AppStarter	2230
6.20.16.1.1	Directory	2231
6.20.16.1.2	Execute	2231
6.20.16.1.3	Icon	2232
6.20.16.1.4	Name	2233
6.20.16.1.5	Params	2233
6.20.16.1.6	Select	2234
6.20.17	Preamp	2234
6.20.18	Preset	2235
6.20.18.1	Channel	2235
6.20.18.1.1	Exec	2235
6.20.18.2	Compatible	2236
6.20.18.3	Exec	2236
6.20.18.4	InputPy	2237
6.20.19	Psa	2237
6.20.19.1	Wideband	2238
6.20.20	Reboot	2238
6.20.21	Revision	2239
6.20.21.1	Factory	2239
6.20.21.2	String	2240
6.20.22	Rsweep	2240
6.20.23	Security	2241
6.20.23.1	State	2241
6.20.24	Sequencer	2242
6.20.25	Set	2243
6.20.26	ShImmediate	2243
6.20.26.1	State	2244
6.20.27	Shutdown	2245
6.20.28	Srecorder	2245
6.20.28.1	Sync	2246
6.20.29	Test	2246
6.20.29.1	Redo	2246
6.20.29.2	Undo	2247
6.21	Trace<Window>	2248
6.21.1	Data	2248
6.21.1.1	Memory	2249
6.21.1.2	X	2250
6.21.2	Iq	2250
6.21.2.1	Data	2251
6.21.2.1.1	Memory	2251
6.21.2.1.2	MemoryAll	2252
6.21.2.2	File	2253
6.21.2.2.1	Repetition	2253
6.21.2.2.1.1	Count	2253
6.21.2.3	Scapture	2254
6.21.2.3.1	Boundary	2254
6.21.2.3.2	Tstamp	2255
6.21.2.3.2.1	Sstart	2255
6.21.2.3.2.2	Trigger	2255
6.22	Trigger	2256

6.22.1	Iq	2256
6.22.1.1	Master	2256
6.22.1.1.1	Source	2257
6.22.1.2	Sender	2257
6.22.1.2.1	Source	2258
6.22.2	Master	2258
6.22.2.1	Port	2259
6.22.3	Sender	2259
6.22.3.1	Port	2260
6.22.4	Sequence	2260
6.22.4.1	BbPower	2261
6.22.4.1.1	Holdoff	2261
6.22.4.2	Dtime	2262
6.22.4.3	Holdoff	2262
6.22.4.3.1	Time	2262
6.22.4.4	IfPower	2263
6.22.4.4.1	Holdoff	2263
6.22.4.4.2	Hysteresis	2264
6.22.4.5	Level	2265
6.22.4.5.1	Am	2265
6.22.4.5.1.1	Absolute	2265
6.22.4.5.1.2	Relative	2266
6.22.4.5.2	External<ExternalPort>	2266
6.22.4.5.3	Fm	2268
6.22.4.5.4	IfPower	2268
6.22.4.5.5	IqPower	2269
6.22.4.5.6	Pm	2270
6.22.4.5.7	RfPower	2270
6.22.4.5.8	Video	2271
6.22.4.6	Oscilloscope	2271
6.22.4.6.1	Coupling	2272
6.22.4.7	RfPower	2272
6.22.4.7.1	Holdoff	2273
6.22.4.8	Slope	2273
6.22.4.9	Source	2274
6.22.4.10	Time	2275
6.22.4.10.1	Rinterval	2275
6.23	TriggerInvoke	2276
6.24	Unit	2276
6.24.1	Angle	2277
6.24.2	Pmeter<PowerMeter>	2277
6.24.2.1	Power	2278
6.24.2.1.1	Ratio	2279
6.24.3	Thd	2280
7	RsFswp Utilities	2281
8	RsFswp Logger	2287
9	RsFswp Events	2291
10	Index	2293
	Index	2295



REVISION HISTORY

1.1 RsFswp

Rohde & Schwarz FSWP Phase Noise Analyzer RsFswp instrument driver.

Basic Hello-World code:

```
from RsFswp import *

instr = RsFswp('TCPIP::192.168.2.101::hislip0')
idn = instr.query('*IDN?')
print('Hello, I am: ' + idn)
```

Check out the full documentation on [ReadTheDocs](#).

Supported instruments: FSWP, FSPN

The package is hosted here: <https://pypi.org/project/RsFswp/>

Examples: https://github.com/Rohde-Schwarz/Examples/tree/main/SpectrumAnalyzers/Python/RsFswp_ScpiPackage

1.1.1 Version history

Latest release notes summary: Updated IQ Analyzer Application commands

Version 3.0.1

- Updated IQ Analyzer Application commands

Version 3.0.0

- Update for FSWP FW 3.0

Version 2.0.1

- Update Documentation

Version 2.0.0

- First released version

GETTING STARTED

2.1 Introduction



RsFswp is a Python remote-control communication module for Rohde & Schwarz SCPI-based Test and Measurement Instruments. It represents SCPI commands as fixed APIs and hence provides SCPI autocompletion and helps you to avoid common string typing mistakes.

Basic example of the idea:

SCPI command:

SYSTem:REFeRence:FREQuency:SOURce

Python module representation:

writing:

`driver.system.reference.frequency.source.set()`

reading:

`driver.system.reference.frequency.source.get()`

Check out this RsFswp example:

```
"""Getting started - how to work with RsFswp Python SCPI package.
This example performs basic RF settings and measurements on an FSW instrument.
It shows the RsFswp calls and their corresponding SCPI commands.
Notice that the python RsFswp interfaces track the SCPI commands syntax."""
```

```
from RsFswp import *

# A good practice is to check for the installed version
RsFswp.assert_minimum_version('2.0.0')

# Open the session
fswp = RsFswp('TCPIP::localhost::HISLIP', reset=True)
# Greetings, stranger...
print(f'Hello, I am: {fswp.utilities.idn_string}')

# Print commands to the console with the logger
fswp.utilities.logger.mode = LoggingMode.On
```

(continues on next page)

(continued from previous page)

```

fswp.utilities.logger.log_to_console = True

# Driver's instrument status checking ( SYST:ERR? ) after each command (default value is_
↳ true):
fswp.utilities.instrument_status_checking = True

#  SYSTem:DISPlay:UPDate ON
fswp.system.display.update.set(True)

#  INITiate:CONTinuous OFF
fswp.initiate.continuous.set(False)
print(f'Always work in single-sweep mode.')

#  SENSE:FREQuency:STARt 1000000000
fswp.sense.frequency.start.set(100E6)

#  SENSE:FREQuency:STOP 2000000000
fswp.sense.frequency.stop.set(200E6)

#  DISPlay:WINDow:TRACe:Y:SCALe:RLEVel -20.0
fswp.display.window.trace.y.scale.refLevel.set(-20.0)

#  DISPlay1:WINDow:SUBWindow:TRACe1:MODE:MAXHold
fswp.display.window.subwindow.trace.mode.set(enums.TraceModeC.MAXHold, repcap.Window.Nr1,
↳ repcap.SubWindow.Default, repcap.Trace.Tr1)

#  DISPlay1: WINDow:SUBWindow:TRACe2:MODE MINHold
fswp.display.window.subwindow.trace.mode.set(enums.TraceModeC.MINHold, repcap.Window.Nr1,
↳ repcap.SubWindow.Default, repcap.Trace.Tr2)

#  SENSE:SWEEp:POINts 10001
fswp.sense.sweep.points.set(10001)

#  INITiate:IMMediate (with timeout 3000 ms)
fswp.initiate.immediate_with_opc(3000)

#          TRACe1:DATA?
trace1 = fswp.trace.data.get(enums.TraceNumber.TRACe1)

#          TRACe2:DATA?
trace2 = fswp.trace.data.get(enums.TraceNumber.TRACe2)

#  CALCulate1:MARKer1:TRACe 1
fswp.calculate.marker.trace.set(1, repcap.Window.Nr1, repcap.Marker.Nr1)

#  CALCulate1:MARKer1:MAXimum:PEAK
fswp.calculate.marker.maximum.peak.set(repcap.Window.Nr1, repcap.Marker.Nr1)
#          CALCulate1:MARKer1:X?
m1x = fswp.calculate.marker.x.get(repcap.Window.Nr1, repcap.Marker.Nr1)

#          CALCulate1:MARKer1:Y?
m1y = fswp.calculate.marker.y.get(repcap.Window.Nr1, repcap.Marker.Nr1)

```

(continues on next page)

(continued from previous page)

```

print(f'Trace 1 points: {len(trace1)}')
print(f'Trace 1 Marker 1: {m1x} Hz, {m1y} dBm')

#   CALCulate1:MARKer2:TRACe 2
fswp.calculate.marker.trace.set(2, repcap.Window.Nr1, repcap.Marker.Nr2)

#   CALCulate1:MARKer2:MINimum:PEAK
fswp.calculate.marker.minimum.peak.set(repcap.Window.Nr1, repcap.Marker.Nr2)

#           CALCulate2:MARKer2:X?
m2x = fswp.calculate.marker.x.get(repcap.Window.Nr1, repcap.Marker.Nr2)

#           CALCulate2:MARKer2:Y?
m2y = fswp.calculate.marker.y.get(repcap.Window.Nr1, repcap.Marker.Nr2)

print(f'Trace 1 points: {len(trace2)}')
print(f'Trace 1 Marker 1: {m2x} Hz, {m2y} dBm')

# Close the session
fswp.close()

```

Couple of reasons why to choose this module over plain SCPI approach:

- Type-safe API using typing module
- You can still use the plain SCPI communication
- You can select which VISA to use or even not use any VISA at all
- Initialization of a new session is straight-forward, no need to set any other properties
- Many useful features are already implemented - reset, self-test, opc-synchronization, error checking, option checking
- Binary data blocks transfer in both directions
- Transfer of arrays of numbers in binary or ASCII format
- File transfers in both directions
- Events generation in case of error, sent data, received data, chunk data (for big files transfer)
- Multithreading session locking - you can use multiple threads talking to one instrument at the same time
- Logging feature tailored for SCPI communication - different for binary and ascii data

2.2 Installation

RsFswp is hosted on pypi.org. You can install it with pip (for example, `pip.exe` for Windows), or if you are using Pycharm (and you should be :-)) direct in the Pycharm Packet Management GUI.

Preconditions

- Installed VISA. You can skip this if you plan to use only socket LAN connection. Download the Rohde & Schwarz VISA for Windows, Linux, Mac OS from [here](#)

Option 1 - Installing with pip.exe under Windows

- Start the command console: WinKey + R, type cmd and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install RsFswp`

Option 2 - Installing in Pycharm

- In Pycharm Menu File->Settings->Project->Project Interpreter click on the '+' button on the top left (the last PyCharm version)
- Type RsFswp in the search box
- If you are behind a Proxy server, configure it in the Menu: File->Settings->Appearance->System Settings->HTTP Proxy

For more information about Rohde & Schwarz instrument remote control, check out our [Instrument Remote Control Web Series](#).

Option 3 - Offline Installation

If you are still reading the installation chapter, it is probably because the options above did not work for you - proxy problems, your boss saw the internet bill... Here are 6 step for installing the RsFswp offline:

- Download this python script (**Save target as**): [rsinstrument_offline_install.py](#) This installs all the preconditions that the RsFswp needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsFswp package to your computer from the pypi.org: <https://pypi.org/project/RsFswp/#files> to for example c:\temp\
- Start the command line WinKey + R, type cmd and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install c:\temp\RsFswp-3.0.1.7.tar`

2.3 Finding Available Instruments

Like the pyvisa's ResourceManager, the RsFswp can search for available instruments:

```
"""
Find the instruments in your environment
"""

from RsFswp import *

# Use the instr_list string items as resource names in the RsFswp constructor
instr_list = RsFswp.list_resources("?*")
print(instr_list)
```

If you have more VISAs installed, the one actually used by default is defined by a secret widget called Visa Conflict Manager. You can force your program to use a VISA of your choice:

```
"""
Find the instruments in your environment with the defined VISA implementation
"""

from RsFswp import *

# In the optional parameter visa_select you can use for example 'rs' or 'ni'
# Rs Visa also finds any NRP-Zxx USB sensors
instr_list = RsFswp.list_resources('?*', 'rs')
print(instr_list)
```

Tip: We believe our R&S VISA is the best choice for our customers. Here are the reasons why:

- Small footprint
- Superior VXI-11 and HiSLIP performance
- Integrated legacy sensors NRP-Zxx support
- Additional VXI-11 and LXI devices search
- Availability for Windows, Linux, Mac OS

2.4 Initiating Instrument Session

RsFswp offers four different types of starting your remote-control session. We begin with the most typical case, and progress with more special ones.

Standard Session Initialization

Initiating new instrument session happens, when you instantiate the RsFswp object. Below, is a simple Hello World example. Different resource names are examples for different physical interfaces.

```
"""
Simple example on how to use the RsFswp module for remote-controlling your instrument
Preconditions:

- Installed RsFswp Python module Version 3.0.1 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsFswp import *

# A good practice is to assure that you have a certain minimum version installed
RsFswp.assert_minimum_version('3.0.1')
resource_string_1 = 'TCPIP::192.168.2.101::INSTR' # Standard LAN connection (also
↳ called VXI-11)
resource_string_2 = 'TCPIP::192.168.2.101::hislip0' # Hi-Speed LAN connection - see
↳ 1MA208
resource_string_3 = 'GPIB::20::INSTR' # GPIB Connection
resource_string_4 = 'USB::0x0AAD::0x0119::022019943::INSTR' # USB-TMC (Test and
↳ Measurement Class)

# Initializing the session
driver = RsFswp(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsFswp package version: {driver.utilities.driver_version}')
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
print(f'Instrument full name: {driver.utilities.full_instrument_model_name}')
print(f'Instrument installed options: {",".join(driver.utilities.instrument_options)}')

# Close the session
driver.close()
```

Note: If you are wondering about the missing ASRL1::INSTR, yes, it works too, but come on... it's 2021.

Do not care about specialty of each session kind; RsFswp handles all the necessary session settings for you. You immediately have access to many identification properties in the interface `driver.utilities`. Here are some of them:

- `idn_string`
- `driver_version`
- `visa_manufacturer`
- `full_instrument_model_name`
- `instrument_serial_number`
- `instrument_firmware_version`

- instrument_options

The constructor also contains optional boolean arguments `id_query` and `reset`:

```
driver = RsFswp('TCPIP::192.168.56.101::HISLIP', id_query=True, reset=True)
```

- Setting `id_query` to `True` (default is `True`) checks, whether your instrument can be used with the `RsFswp` module.
- Setting `reset` to `True` (default is `False`) resets your instrument. It is equivalent to calling the `reset()` method.

Selecting a Specific VISA

Just like in the function `list_resources()`, the `RsFswp` allows you to choose which VISA to use:

```
"""
Choosing VISA implementation
"""

from RsFswp import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsFswp('TCPIP::192.168.56.101::INSTR', True, True, "SelectVisa='rs'")

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f"\nI am using the VISA from: {driver.utilities.visa_manufacturer}")

# Close the session
driver.close()
```

No VISA Session

We recommend using VISA when possible preferably with HiSlip session because of its low latency. However, if you are a strict VISA denier, `RsFswp` has something for you too - **no Visa installation raw LAN socket**:

```
"""
Using RsFswp without VISA for LAN Raw socket communication
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.56.101::5025::SOCKET', True, True, "SelectVisa='socket'")
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
print(f"\nHello, I am: '{driver.utilities.idn_string}'")

# Close the session
driver.close()
```

Warning: Not using VISA can cause problems by debugging when you want to use the communication Trace Tool. The good news is, you can easily switch to use VISA and back just by changing the constructor arguments. The rest of your code stays unchanged.

Simulating Session

If a colleague is currently occupying your instrument, leave him in peace, and open a simulating session:

```
driver = RsFswp('TCPIP::192.168.56.101::HISLIP', True, True, "Simulate=True")
```

More option_string tokens are separated by comma:

```
driver = RsFswp('TCPIP::192.168.56.101::HISLIP', True, True, "SelectVisa='rs',  
↳ Simulate=True")
```

Shared Session

In some scenarios, you want to have two independent objects talking to the same instrument. Rather than opening a second VISA connection, share the same one between two or more RsFswp objects:

```
"""  
Sharing the same physical VISA session by two different RsFswp objects  
"""  
  
from RsFswp import *  
  
driver1 = RsFswp('TCPIP::192.168.56.101::INSTR', True, True)  
driver2 = RsFswp.from_existing_session(driver1)  
  
print(f'driver1: {driver1.utilities.idn_string}')  
print(f'driver2: {driver2.utilities.idn_string}')  
  
# Closing the driver2 session does not close the driver1 session - driver1 is the  
↳ 'session master'  
driver2.close()  
print(f'driver2: I am closed now')  
  
print(f'driver1: I am still opened and working: {driver1.utilities.idn_string}')  
driver1.close()  
print(f'driver1: Only now I am closed.')
```

Note: The driver1 is the object holding the ‘master’ session. If you call the driver1.close(), the driver2 loses its instrument session as well, and becomes pretty much useless.

2.5 Plain SCPI Communication

After you have opened the session, you can use the instrument-specific part described in the RsFswp API Structure. If for any reason you want to use the plain SCPI, use the utilities interface’s two basic methods:

- write_str() - writing a command without an answer, for example *RST
- query_str() - querying your instrument, for example the *IDN? query

You may ask a question. Actually, two questions:

- Q1: Why there are not called write() and query() ?

- **Q2:** Where is the `read()` ?

Answer 1: Actually, there are - the `write_str()` / `write()` and `query_str()` / `query()` are aliases, and you can use any of them. We promote the `_str` names, to clearly show you want to work with strings. Strings in Python3 are Unicode, the *bytes* and *string* objects are not interchangeable, since one character might be represented by more than 1 byte. To avoid mixing string and binary communication, all the method names for binary transfer contain `_bin` in the name.

Answer 2: Short answer - you do not need it. Long answer - your instrument never sends unsolicited responses. If you send a set command, you use `write_str()`. For a query command, you use `query_str()`. So, you really do not need it...

Bottom line - if you are used to `write()` and `query()` methods, from `pyvisa`, the `write_str()` and `query_str()` are their equivalents.

Enough with the theory, let us look at an example. Simple write, and query:

```
"""
Basic string write_str / query_str
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.56.101::INSTR')
driver.utilities.write_str('*RST')
response = driver.utilities.query_str('*IDN?')
print(response)

# Close the session
driver.close()
```

This example is so-called “*University-Professor-Example*” - good to show a principle, but never used in praxis. The abovementioned commands are already a part of the driver’s API. Here is another example, achieving the same goal:

```
"""
Basic string write_str / query_str
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.56.101::INSTR')
driver.utilities.reset()
print(driver.utilities.idn_string)

# Close the session
driver.close()
```

One additional feature we need to mention here: **VISA timeout**. To simplify, VISA timeout plays a role in each `query_xxx()`, where the controller (your PC) has to prevent waiting forever for an answer from your instrument. VISA timeout defines that maximum waiting time. You can set/read it with the `visa_timeout` property:

```
# Timeout in milliseconds
driver.utilities.visa_timeout = 3000
```

After this time, the `RsFswp` raises an exception. Speaking of exceptions, an important feature of the `RsFswp` is **Instrument Status Checking**. Check out the next chapter that describes the error checking in details.

For completion, we mention other string-based `write_xxx()` and `query_xxx()` methods - all in one example. They are convenient extensions providing type-safe float/boolean/integer setting/querying features:

```
"""
Basic string write_xxx / query_xxx
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.56.101::INSTR')
driver.utilities.visa_timeout = 5000
driver.utilities.instrument_status_checking = True
driver.utilities.write_int('SWEEP:COUNT ', 10) # sending 'SWEEP:COUNT 10'
driver.utilities.write_bool('SOURCE:RF:OUTPUT:STATE ', True) # sending
↳ 'SOURCE:RF:OUTPUT:STATE ON'
driver.utilities.write_float('SOURCE:RF:FREQUENCY ', 1E9) # sending 'SOURCE:RF:FREQUENCY_
↳ 1000000000'

sc = driver.utilities.query_int('SWEEP:COUNT?') # returning integer number sc=10
out = driver.utilities.query_bool('SOURCE:RF:OUTPUT:STATE?') # returning boolean_
↳ out=True
freq = driver.utilities.query_float('SOURCE:RF:FREQUENCY?') # returning float number_
↳ freq=1E9

# Close the session
driver.close()
```

Lastly, a method providing basic synchronization: `query_opc()`. It sends query `*OPC?` to your instrument. The instrument waits with the answer until all the tasks it currently has in a queue are finished. This way your program waits too, and this way it is synchronized with the actions in the instrument. Remember to have the VISA timeout set to an appropriate value to prevent the timeout exception. Here's the snippet:

```
driver.utilities.visa_timeout = 3000
driver.utilities.write_str("INIT")
driver.utilities.query_opc()

# The results are ready now to fetch
results = driver.utilities.query_str("FETCH:MEASUREMENT?")
```

Tip: Wait, there's more: you can send the `*OPC?` after each `write_xxx()` automatically:

```
# Default value after init is False
driver.utilities.opc_query_after_write = True
```

2.6 Error Checking

RsFswp pushes limits even further (internal R&S joke): It has a built-in mechanism that after each command/query checks the instrument's status subsystem, and raises an exception if it detects an error. For those who are already screaming: **Speed Performance Penalty!!!**, don't worry, you can disable it.

Instrument status checking is very useful since in case your command/query caused an error, you are immediately informed about it. Status checking has in most cases no practical effect on the speed performance of your program. However, if for example, you do many repetitions of short write/query sequences, it might make a difference to switch it off:

```
# Default value after init is True
driver.utilities.instrument_status_checking = False
```

To clear the instrument status subsystem of all errors, call this method:

```
driver.utilities.clear_status()
```

Instrument's status system error queue is clear-on-read. It means, if you query its content, you clear it at the same time. To query and clear list of all the current errors, use this snippet:

```
errors_list = driver.utilities.query_all_errors()
```

See the next chapter on how to react on errors.

2.7 Exception Handling

The base class for all the exceptions raised by the RsFswp is `RsInstrException`. Inherited exception classes:

- `ResourceError` raised in the constructor by problems with initiating the instrument, for example wrong or non-existing resource name
- `StatusException` raised if a command or a query generated error in the instrument's error queue
- `TimeoutException` raised if a visa timeout or an opc timeout is reached

In this example we show usage of all of them. Because it is difficult to generate an error using the instrument-specific SCPI API, we use plain SCPI commands:

```
"""
Showing how to deal with exceptions
"""

from RsFswp import *

driver = None
# Try-catch for initialization. If an error occurs, the ResourceError is raised
try:
    driver = RsFswp('TCPIP::10.112.1.179::HISLIP')
except ResourceError as e:
    print(e.args[0])
    print('Your instrument is probably OFF...')
    # Exit now, no point of continuing
    exit(1)
```

(continues on next page)

(continued from previous page)

```
# Dealing with commands that potentially generate errors OPTION 1:
# Switching the status checking OFF temporarily
driver.utilities.instrument_status_checking = False
driver.utilities.write_str('MY:MISSpelled:COMManD')
# Clear the error queue
driver.utilities.clear_status()
# Status checking ON again
driver.utilities.instrument_status_checking = True

# Dealing with queries that potentially generate errors OPTION 2:
try:
    # You might want to reduce the VISA timeout to avoid long waiting
    driver.utilities.visa_timeout = 1000
    driver.utilities.query_str('MY:WRONG:QUERy?')

except StatusException as e:
    # Instrument status error
    print(e.args[0])
    print('Nothing to see here, moving on...')

except TimeoutException as e:
    # Timeout error
    print(e.args[0])
    print('That took a long time...')

except RsInstrException as e:
    # RsInstrException is a base class for all the RsFswp exceptions
    print(e.args[0])
    print('Some other RsFswp error...')

finally:
    driver.utilities.visa_timeout = 5000
    # Close the session in any case
    driver.close()
```

Tip: General rules for exception handling:

- If you are sending commands that might generate errors in the instrument, for example deleting a file which does not exist, use the **OPTION 1** - temporarily disable status checking, send the command, clear the error queue and enable the status checking again.
 - If you are sending queries that might generate errors or timeouts, for example querying measurement that can not be performed at the moment, use the **OPTION 2** - try/except with optionally adjusting the timeouts.
-

2.8 Transferring Files

Instrument -> PC

You definitely experienced it: you just did a perfect measurement, saved the results as a screenshot to an instrument's storage drive. Now you want to transfer it to your PC. With RsFswp, no problem, just figure out where the screenshot was stored on the instrument. In our case, it is `/var/user/instr_screenshot.png`:

```
driver.utilities.read_file_from_instrument_to_pc(
    r'/var/user/instr_screenshot.png',
    r'c:\temp\pc_screenshot.png')
```

PC -> Instrument

Another common scenario: Your cool test program contains a setup file you want to transfer to your instrument: Here is the RsFswp one-liner split into 3 lines:

```
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\instr_setup.sav',
    r'/var/appdata/instr_setup.sav')
```

2.9 Writing Binary Data

Writing from bytes

An example where you need to send binary data is a waveform file of a vector signal generator. First, you compose your `wform_data` as bytes, and then you send it with `write_bin_block()`:

```
# MyWaveform.wv is an instrument file name under which this data is stored
driver.utilities.write_bin_block(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv'",",
    wform_data)
```

Note: Notice the `write_bin_block()` has two parameters:

- string parameter `cmd` for the SCPI command
- bytes parameter `payload` for the actual binary data to send

Writing from PC files

Similar to querying binary data to a file, you can write binary data from a file. The second parameter is then the PC file path the content of which you want to send:

```
driver.utilities.write_bin_block_from_file(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv'",",
    r"c:\temp\wform_data.wv")
```

2.10 Transferring Big Data with Progress

We can agree that it can be annoying using an application that shows no progress for long-lasting operations. The same is true for remote-control programs. Luckily, the RsFswp has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsFswp allows you to register a function (programmers fancy name is `callback`), which is then periodically invoked after transfer of one data chunk. You can define that chunk size, which gives you control over the callback invoke frequency. You can even slow down the transfer speed, if you want to process the data as they arrive (direction instrument -> PC).

To show this in praxis, we are going to use another *University-Professor-Example*: querying the `*IDN?` with chunk size of 2 bytes and delay of 200ms between each chunk read:

```
"""
Event handlers by reading
"""

from RsFswp import *
import time

def my_transfer_handler(args):
    """Function called each time a chunk of data is transferred"""
    # Total size is not always known at the beginning of the transfer
    total_size = args.total_size if args.total_size is not None else "unknown"

    print(f"Context: '{args.context}{'with opc' if args.opc_sync else ''}', "
          f"chunk {args.chunk_ix}, "
          f"transferred {args.transferred_size} bytes, "
          f"total size {total_size}, "
          f"direction {'reading' if args.reading else 'writing'}, "
          f"data '{args.data}'")

    if args.end_of_transfer:
        print('End of Transfer')
        time.sleep(0.2)

driver = RsFswp('TCPIP::192.168.56.101::INSTR')

driver.events.on_read_handler = my_transfer_handler
# Switch on the data to be included in the event arguments
# The event arguments args.data will be updated
driver.events.io_events_include_data = True
# Set data chunk size to 2 bytes
driver.utilities.data_chunk_size = 2
driver.utilities.query_str('*IDN?')
# Unregister the event handler
driver.utilities.on_read_handler = None

# Close the session
driver.close()
```

If you start it, you might wonder (or maybe not): why is the `args.total_size = None`? The reason is, in this particular case the RsFswp does not know the size of the complete response up-front. However, if you use the same mechanism for transfer of a known data size (for example, file transfer), you get the information about the total size too, and hence you can calculate the progress as:

$$progress [pct] = 100 * args.transferred_size / args.total_size$$

Snippet of transferring file from PC to instrument, the rest of the code is the same as in the previous example:

```
driver.events.on_write_handler = my_transfer_handler
driver.events.io_events_include_data = True
driver.data_chunk_size = 10000
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\my_big_file.bin',
    r'/var/user/my_big_file.bin')
# Unregister the event handler
driver.events.on_write_handler = None
```

2.11 Multithreading

You are at the party, many people talking over each other. Not every person can deal with such crosstalk, neither can measurement instruments. For this reason, RsFswp has a feature of scheduling the access to your instrument by using so-called **Locks**. Locks make sure that there can be just one client at a time *talking* to your instrument. Talking in this context means completing one communication step - one command write or write/read or write/read/error check.

To describe how it works, and where it matters, we take three typical multithread scenarios:

One instrument session, accessed from multiple threads

You are all set - the lock is a part of your instrument session. Check out the following example - it will execute properly, although the instrument gets 10 queries at the same time:

```
"""
Multiple threads are accessing one RsFswp object
"""

import threading
from RsFswp import *

def execute(session):
    """Executed in a separate thread."""
    session.utilities.query_str('*IDN?')

driver = RsFswp('TCPIP::192.168.56.101::INSTR')
threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver, ))
    t.start()
    threads.append(t)
print('All threads started')
```

(continues on next page)

(continued from previous page)

```

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver.close()

```

Shared instrument session, accessed from multiple threads

Same as the previous case, you are all set. The session carries the lock with it. You have two objects, talking to the same instrument from multiple threads. Since the instrument session is shared, the same lock applies to both objects causing the exclusive access to the instrument.

Try the following example:

```

"""
Multiple threads are accessing two RsFswp objects with shared session
"""

import threading
from RsFswp import *

def execute(session: RsFswp, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')

driver1 = RsFswp('TCPIP::192.168.56.101::INSTR')
driver2 = RsFswp.from_existing_session(driver1)
driver1.utilities.visa_timeout = 200
driver2.utilities.visa_timeout = 200
# To see the effect of crosstalk, uncomment this line
# driver2.utilities.clear_lock()

threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()

```

(continues on next page)

(continued from previous page)

```
print('All threads ended')

driver2.close()
driver1.close()
```

As you see, everything works fine. If you want to simulate some party crosstalk, uncomment the line `driver2.utilities.clear_lock()`. This causes the driver2 session lock to break away from the driver1 session lock. Although the driver1 still tries to schedule its instrument access, the driver2 tries to do the same at the same time, which leads to all the fun stuff happening.

Multiple instrument sessions accessed from multiple threads

Here, there are two possible scenarios depending on the instrument's VISA interface:

- You are lucky, because your instrument handles each remote session completely separately. An example of such instrument is SMW200A. In this case, you have no need for session locking.
- Your instrument handles all sessions with one set of in/out buffers. You need to lock the session for the duration of a talk. And you are lucky again, because the RsFswp takes care of it for you. The text below describes this scenario.

Run the following example:

```
"""
Multiple threads are accessing two RsFswp objects with two separate sessions
"""

import threading
from RsFswp import *

def execute(session: RsFswp, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')

driver1 = RsFswp('TCPIP::192.168.56.101::INSTR')
driver2 = RsFswp('TCPIP::192.168.56.101::INSTR')
driver1.utilities.visa_timeout = 200
driver2.utilities.visa_timeout = 200

# Synchronise the sessions by sharing the same lock
driver2.utilities.assign_lock(driver1.utilities.get_lock()) # To see the effect of
↳ crosstalk, comment this line

threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
```

(continues on next page)

(continued from previous page)

```

    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver2.close()
driver1.close()

```

You have two completely independent sessions that want to talk to the same instrument at the same time. This will not go well, unless they share the same session lock. The key command to achieve this is `driver2.utilities.assign_lock(driver1.utilities.get_lock())`. Try to comment it and see how it goes. If despite commenting the line the example runs without issues, you are lucky to have an instrument similar to the SMW200A.

2.12 Logging

Yes, the logging again. This one is tailored for instrument communication. You will appreciate such handy feature when you troubleshoot your program, or just want to protocol the SCPI communication for your test reports.

What can you actually do with the logger?

- Write SCPI communication to a stream-like object, for example console or file, or both simultaneously
- Log only errors and skip problem-free parts; this way you avoid going through thousands lines of texts
- Investigate duration of certain operations to optimize your program's performance
- Log custom messages from your program

Let us take this basic example:

```

"""
Basic logging example to the console
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.1.101::INSTR')

# Switch ON logging to the console.
driver.utilities.logger.log_to_console = True
driver.utilities.logger.mode = LoggingMode.On
driver.utilities.reset()

# Close the session
driver.close()

```

Console output:

```

10:29:10.819    TCPIP::192.168.1.101::INSTR    0.976 ms  Write: *RST
10:29:10.819    TCPIP::192.168.1.101::INSTR 1884.985 ms  Status check: OK

```

(continues on next page)

(continued from previous page)

10:29:12.704	TCPIP::192.168.1.101::INSTR	0.983 ms	Query OPC: 1
10:29:12.705	TCPIP::192.168.1.101::INSTR	2.892 ms	Clear status: OK
10:29:12.708	TCPIP::192.168.1.101::INSTR	3.905 ms	Status check: OK
10:29:12.712	TCPIP::192.168.1.101::INSTR	1.952 ms	Close: Closing session

The columns of the log are aligned for better reading. Columns meaning:

- (1) Start time of the operation
- (2) Device resource name (you can set an alias)
- (3) Duration of the operation
- (4) Log entry

Tip: You can customize the logging format with `set_format_string()`, and set the maximum log entry length with the properties:

- `abbreviated_max_len_ascii`
- `abbreviated_max_len_bin`
- `abbreviated_max_len_list`

See the full logger help [here](#).

Notice the SCPI communication starts from the line `driver.utilities.reset()`. If you want to log the initialization of the session as well, you have to switch the logging ON already in the constructor:

```
driver = RsFswp('TCPIP::192.168.56.101::HISLIP', options='LoggingMode=On')
```

Parallel to the console logging, you can log to a general stream. Do not fear the programmer's jargon... under the term **stream** you can just imagine a file. To be a little more technical, a stream in Python is any object that has two methods: `write()` and `flush()`. This example opens a file and sets it as logging target:

```
"""
Example of logging to a file
"""

from RsFswp import *

driver = RsFswp('TCPIP::192.168.1.101::INSTR')

# We also want to log to the console.
driver.utilities.logger.log_to_console = True

# Logging target is our file
file = open(r'c:\temp\my_file.txt', 'w')
driver.utilities.logger.set_logging_target(file)
driver.utilities.logger.mode = LoggingMode.On

# Instead of the 'TCPIP::192.168.1.101::INSTR', show 'MyDevice'
driver.utilities.logger.device_name = 'MyDevice'

# Custom user entry
```

(continues on next page)

(continued from previous page)

```
driver.utilities.logger.info_raw('----- This is my custom log entry. ---- ')\n\ndriver.utilities.reset()\n\n# Close the session\ndriver.close()\n\n# Close the log file\nfile.close()
```

Tip: To make the log more compact, you can skip all the lines with Status check: OK:

```
driver.utilities.logger.log_status_check_ok = False
```

Hint: You can share the logging file between multiple sessions. In such case, remember to close the file only after you have stopped logging in all your sessions, otherwise you get a log write error.

For logging to a UDP port in addition to other log targets, use one of the lines:

```
driver.utilities.logger.log_to_udp = True\ndriver.utilities.logger.log_to_console_and_udp = True
```

You can select the UDP port to log to, the default is 49200:

```
driver.utilities.logger.udp_port = 49200
```

Another cool feature is logging only errors. To make this mode usefull for troubleshooting, you also want to see the circumstances which lead to the errors. Each driver elementary operation, for example, `write_str()`, can generate a group of log entries - let us call them **Segment**. In the logging mode **Errors**, a whole segment is logged only if at least one entry of the segment is an error.

The script below demonstrates this feature. We use a direct SCPI communication to send a misspelled SCPI command ***CLS**, which leads to instrument status error:

```
"""\nLogging example to the console with only errors logged\n"""\n\nfrom RsFswp import *\n\ndriver = RsFswp('TCPIP::192.168.1.101::INSTR', options='LoggingMode=Errors')\n\n# Switch ON logging to the console.\ndriver.utilities.logger.log_to_console = True\n\n# Reset will not be logged, since no error occurred there\ndriver.utilities.reset()\n\n# Now a misspelled command.\ndriver.utilities.write('*CLaS')
```

(continues on next page)

(continued from previous page)

```
# A good command again, no logging here
idn = driver.utilities.query('*IDN?')

# Close the session
driver.close()
```

Console output:

```
12:11:02.879 TCPIP::192.168.1.101::INSTR    0.976 ms Write string: *CLaS
12:11:02.879 TCPIP::192.168.1.101::INSTR    6.833 ms Status check: StatusException:
                                     Instrument error detected: Undefined header;
↪ *CLaS
```

Notice the following:

- Although the operation **Write string: *CLaS** finished without an error, it is still logged, because it provides the context for the actual error which occurred during the status checking right after.
- No other log entries are present, including the session initialization and close, because they were all error-free.

3.1 AccessType

```
# Example value:  
value = enums.AccessType.RO  
# All values (2x):  
RO | RW
```

3.2 AdcPreFilterMode

```
# Example value:  
value = enums.AdcPreFilterMode.AUTO  
# All values (2x):  
AUTO | WIDE
```

3.3 AdemMeasType

```
# Example value:  
value = enums.AdemMeasType.MIDDLE  
# All values (5x):  
MIDDLE | MPEak | NPEak | PPEak | RMS
```

3.4 AdjustAlignment

```
# Example value:  
value = enums.AdjustAlignment.CENTer  
# All values (3x):  
CENTer | LEFT | RIGHT
```

3.5 AngleUnit

```
# Example value:  
value = enums.AngleUnit.DEG  
# All values (2x):  
DEG | RAD
```

3.6 AnnotationMode

```
# Example value:  
value = enums.AnnotationMode.CSPan  
# All values (2x):  
CSPan | SSTop
```

3.7 AttenuatorMode

```
# Example value:  
value = enums.AttenuatorMode.LDISortion  
# All values (3x):  
LDISortion | LNOise | NORMal
```

3.8 AutoManualMode

```
# Example value:  
value = enums.AutoManualMode.AUTO  
# All values (2x):  
AUTO | MANual
```

3.9 AutoManualUserMode

```
# Example value:  
value = enums.AutoManualUserMode.AUTO  
# All values (3x):  
AUTO | MANual | USER
```

3.10 AutoMode

```
# Example value:
value = enums.AutoMode.AUTO
# All values (3x):
AUTO | OFF | ON
```

3.11 AutoOrOff

```
# Example value:
value = enums.AutoOrOff.AUTO
# All values (2x):
AUTO | OFF
```

3.12 AverageModeA

```
# Example value:
value = enums.AverageModeA.LINEar
# All values (3x):
LINEar | LOGarithmic | POWer
```

3.13 AverageModeB

```
# Example value:
value = enums.AverageModeB.LINEar
# All values (6x):
LINEar | LOGarithmic | MAXimum | POWer | SCALar | VIDEO
```

3.14 Band

```
# First value:
value = enums.Band.A
# Last value:
value = enums.Band.Y
# All values (14x):
A | D | E | F | G | J | K | KA
Q | U | USER | V | W | Y
```

3.15 BandB

```
# First value:
value = enums.BandB.D
# Last value:
value = enums.BandB.Y
# All values (12x):
D | E | F | G | J | KA | Q | U
USER | V | W | Y
```

3.16 BbInputSource

```
# Example value:
value = enums.BbInputSource.AIQ
# All values (4x):
AIQ | DIQ | FIQ | RF
```

3.17 BerRateFormat

```
# First value:
value = enums.BerRateFormat.CURRENT
# Last value:
value = enums.BerRateFormat.TTOTA
# All values (17x):
CURRENT | DSINDEX | MAX | MIN | SECURRENT | SEMAX | SEMIN | SETOTAL
TCURRENT | TECURRENT | TEMAX | TEMIN | TETOTAL | TMAX | TMIN | TOTAl
TTOTAL
```

3.18 BitOrdering

```
# Example value:
value = enums.BitOrdering.LSB
# All values (2x):
LSB | MSB
```

3.19 BurstMode

```
# Example value:
value = enums.BurstMode.BURS
# All values (2x):
BURS | MEAS
```

3.20 CalibrationScope

```
# Example value:
value = enums.CalibrationScope.AClearn
# All values (4x):
AClearn | ALL | OFF | ON
```

3.21 CalibrationState

```
# Example value:
value = enums.CalibrationState.EXpired
# All values (3x):
EXpired | NAN | OK
```

3.22 ChannelType

```
# First value:
value = enums.ChannelType.IqAnalyzer=IQ
# Last value:
value = enums.ChannelType.SpectrumAnalyzer=SANALYZER
# All values (31x):
IqAnalyzer | K10_Gsm | K106_NbIot | K10x_Lte | K118_Verizon5G | K14x_5GnewRadio | K15_
↳ Avionics | K17_MultiCarrierGroupDelay
K18_AmplifierMeas | K192_193_Docsis31 | K201_OneWeb | K30_Noise | K40_PhaseNoise | K50_
↳ FastSpurSearch | K6_PulseAnalysis | K60_TransientAnalysis
K7_AnalogModulation | K70_VectorSignalAnalyzer | K72_3GppFddBts | K73_3GppFddUe | K76_
↳ TdScdmaBts | K77_TdScdmaUe | K82_Cdma2000Bts | K83_Cdma2000Ms
K84_EvdoBts | K85_EvdoMs | K91_Wlan | K95_80211ad | K97_80211ay | RealTimeSpectrum |
↳ SpectrumAnalyzer
```

3.23 CheckResult

```
# Example value:
value = enums.CheckResult.FAILED
# All values (2x):
FAILED | PASSED
```

3.24 Color

```
# First value:
value = enums.Color.BLACK
# Last value:
value = enums.Color.YELLOW
# All values (16x):
BLACK | BLUE | BROWN | CYAN | DGRAY | GREEN | LBLUE | LCYAN
LGRAY | LGREEN | LMAGENTA | LRED | MAGENTA | RED | WHITE | YELLOW
```

3.25 ColorSchemeA

```
# Example value:
value = enums.ColorSchemeA.COLD
# All values (5x):
COLD | COLOR | GRAYSCALE | HOT | RADAR
```

3.26 CompatibilityMode

```
# First value:
value = enums.CompatibilityMode.ATT
# Last value:
value = enums.CompatibilityMode.FSWXv1_0
# All values (11x):
ATT | DEFAULT | FSET | FSL | FSMR | FSP | FSQ | FSU
FSV | FSW | FSWXv1_0
```

3.27 ComponentType

```
# Example value:
value = enums.ComponentType.AMPLIFIER
# All values (4x):
AMPLIFIER | DIVIDER | MIXER | MULTIPLIER
```


3.28 ConfigMode

```
# Example value:  
value = enums.ConfigMode.DEFAULT  
# All values (2x):  
DEFAULT | USER
```

3.29 CorrectionMeasType

```
# Example value:  
value = enums.CorrectionMeasType.OPEN  
# All values (2x):  
OPEN | THROUGH
```

3.30 CorrectionMethod

```
# Example value:  
value = enums.CorrectionMethod.REFLEXION  
# All values (2x):  
REFLEXION | TRANSMISSION
```

3.31 CorrectionMode

```
# Example value:  
value = enums.CorrectionMode.SPOT  
# All values (2x):  
SPOT | TABLE
```

3.32 Counter

```
# Example value:  
value = enums.Counter.CAPTure  
# All values (2x):  
CAPTure | STATistics
```

3.33 CouplingTypeA

```
# Example value:  
value = enums.CouplingTypeA.AC  
# All values (2x):  
AC | DC
```

3.34 CouplingTypeB

```
# Example value:  
value = enums.CouplingTypeB.AC  
# All values (3x):  
AC | DC | DCLimit
```

3.35 DataExportMode

```
# Example value:  
value = enums.DataExportMode.RAW  
# All values (2x):  
RAW | TRACe
```

3.36 DataFormat

```
# First value:  
value = enums.DataFormat.ASCii  
# Last value:  
value = enums.DataFormat.UINT_cma_64  
# All values (10x):  
ASCii | MATLAB_cma_16 | MATLAB_cma_32 | MATLAB_cma_64 | Real16 | Real32 | Real64 | UINT_  
↪cma_16  
UINT_cma_32 | UINT_cma_64
```

3.37 DaysOfWeek

```
# Example value:  
value = enums.DaysOfWeek.ALL  
# All values (8x):  
ALL | FRIDay | MONDay | SATurday | SUNday | THURsday | TUESday | WEDNesday
```

3.38 DdemGroup

```
# Example value:
value = enums.DdemGroup.APSK
# All values (8x):
APSK | ASK | FSK | MSK | PSK | QAM | QPSK | UQAM
```

3.39 DdemodFilter

```
# First value:
value = enums.DdemodFilter.A25Fm
# Last value:
value = enums.DdemodFilter.RRCosine
# All values (13x):
A25Fm | B22 | B25 | B44 | EMES | EREF | GAUSSian | QFM
QFR | QRM | QRR | RCOSine | RRCosine
```

3.40 DdemResultType

```
# First value:
value = enums.DdemResultType.ADR
# Last value:
value = enums.DdemResultType.RHO
# All values (20x):
ADR | DEV | DTTS | EVPK | EVPS | EVRM | FEPK | FERR
FSPK | FSPS | FSRM | IQIM | IQOF | MEPK | MEPS | MERM
PEPK | PEPS | PERM | RHO
```

3.41 DdemSignalType

```
# Example value:
value = enums.DdemSignalType.BURSted
# All values (2x):
BURSted | CONTinuous
```

3.42 Detect

```
# Example value:
value = enums.Detect.DETected
# All values (2x):
DETECTED | NDETECTED
```

3.43 DetectorB

```
# First value:  
value = enums.DetectorB.ACSine  
# Last value:  
value = enums.DetectorB.SMP  
# All values (13x):  
ACSine | ACVideo | APEak | AVERage | CAVerage | CRMS | NEGative | NRM  
POSitive | QPEak | RMS | SAMPlE | SMP
```

3.44 DetectorC

```
# Example value:  
value = enums.DetectorC.ACSine  
# All values (8x):  
ACSine | ACVideo | APEak | AVERage | NEGative | POSitive | RMS | SAMPlE
```

3.45 DiagnosticSignal

```
# Example value:  
value = enums.DiagnosticSignal.AIQ  
# All values (8x):  
AIQ | CALibration | EMI | MCALibration | RF | SYNThtwo | TG | WBCal
```

3.46 DiqUnit

```
# Example value:  
value = enums.DiqUnit.AMPere  
# All values (8x):  
AMPere | DBM | DBMV | DBPW | DBUA | DBUV | VOLT | WATT
```

3.47 DisplayFormat

```
# Example value:  
value = enums.DisplayFormat.SINGle  
# All values (2x):  
SINGle | SPLit
```

3.48 DisplayPosition

```
# Example value:  
value = enums.DisplayPosition.BOTTom  
# All values (3x):  
BOTTom | OFF | TOP
```

3.49 Duration

```
# Example value:  
value = enums.Duration.LONG  
# All values (3x):  
LONG | NORMAl | SHORt
```

3.50 DutType

```
# Example value:  
value = enums.DutType.AMPLifier  
# All values (4x):  
AMPLifier | DDOWnconv | DOWNconv | UPConv
```

3.51 EgateType

```
# Example value:  
value = enums.EgateType.EDGE  
# All values (2x):  
EDGE | LEVe1
```

3.52 EnrType

```
# Example value:  
value = enums.EnrType.DIODE  
# All values (3x):  
DIODE | RESistor | SMART
```

3.53 EspectrumRtype

```
# Example value:  
value = enums.EspectrumRtype.CPOwer  
# All values (2x):  
CPOwer | PEAK
```

3.54 EventOnce

```
# Example value:  
value = enums.EventOnce.ONCE  
# All values (1x):  
ONCE
```

3.55 EvmCalc

```
# Example value:  
value = enums.EvmCalc.MACPower  
# All values (4x):  
MACPower | MECPower | SIGNal | SYMBol
```

3.56 Factory

```
# Example value:  
value = enums.Factory.ALL  
# All values (3x):  
ALL | PATtern | STANDard
```

3.57 FftFilterMode

```
# Example value:  
value = enums.FftFilterMode.AUTO  
# All values (3x):  
AUTO | NARRow | WIDE
```

3.58 FftWindowType

```
# Example value:
value = enums.FftWindowType.BLACKharris
# All values (8x):
BLACKharris | FLATtop | GAUSSsian | HAMMING | HANNing | KAISerbessel | P5 | RECTangular
```

3.59 FileFormat

```
# Example value:
value = enums.FileFormat.CSV
# All values (2x):
CSV | DAT
```

3.60 FileFormatDdem

```
# Example value:
value = enums.FileFormatDdem.FRES
# All values (2x):
FRES | VAE
```

3.61 FileSeparator

```
# Example value:
value = enums.FileSeparator.COMMa
# All values (3x):
COMMa | SEMicolon | TAB
```

3.62 FilterTypeA

```
# Example value:
value = enums.FilterTypeA.FLAT
# All values (2x):
FLAT | GAUSS
```

3.63 FilterTypeB

```
# First value:  
value = enums.FilterTypeB.CFILter  
# Last value:  
value = enums.FilterTypeB.RRC  
# All values (9x):  
CFILter | CISPr | FFT | MIL | NOISe | NORMal | P5 | PULSe  
RRC
```

3.64 FilterTypeC

```
# Example value:  
value = enums.FilterTypeC.CFILter  
# All values (7x):  
CFILter | CISPr | MIL | NORMal | P5 | PULSe | RRC
```

3.65 FilterTypeK91

```
# Example value:  
value = enums.FilterTypeK91.CFILter  
# All values (5x):  
CFILter | NORMal | P5 | PULSe | RRC
```

3.66 FineSync

```
# Example value:  
value = enums.FineSync.DDATa  
# All values (3x):  
DDATa | KDATa | PATtern
```

3.67 FpeaksSortMode

```
# Example value:  
value = enums.FpeaksSortMode.X  
# All values (2x):  
X | Y
```


3.68 FrameModulation

```
# Example value:  
value = enums.FrameModulation.AUTO  
# All values (3x):  
AUTO | DATA | PATtern
```

3.69 FrameModulationB

```
# Example value:  
value = enums.FrameModulationB.DATA  
# All values (2x):  
DATA | PATtern
```

3.70 FramesScope

```
# Example value:  
value = enums.FramesScope.ALL  
# All values (2x):  
ALL | CHANnel
```

3.71 FrequencyCouplingLinkA

```
# Example value:  
value = enums.FrequencyCouplingLinkA.OFF  
# All values (3x):  
OFF | RBW | SPAN
```

3.72 FrequencyType

```
# Example value:  
value = enums.FrequencyType.IF  
# All values (3x):  
IF | LO | RF
```

3.73 FunctionA

```
# Example value:  
value = enums.FunctionA.MAX  
# All values (2x):  
MAX | OFF
```

3.74 FunctionB

```
# Example value:  
value = enums.FunctionB.MAX  
# All values (3x):  
MAX | NONE | SUM
```

3.75 GatedSourceK30

```
# Example value:  
value = enums.GatedSourceK30.EXT2  
# All values (3x):  
EXT2 | EXT3 | EXTERNAL
```

3.76 GeneratorIntf

```
# Example value:  
value = enums.GeneratorIntf.GPIB  
# All values (2x):  
GPIB | TCPip
```

3.77 GeneratorIntfType

```
# Example value:  
value = enums.GeneratorIntfType.GPIB  
# All values (3x):  
GPIB | PEXpress | TCPip
```

3.78 GeneratorLink

```
# Example value:  
value = enums.GeneratorLink.GPIB  
# All values (2x):  
GPIB | TTL
```

3.79 GpibTerminator

```
# Example value:  
value = enums.GpibTerminator.EOI  
# All values (2x):  
EOI | LFEoi
```

3.80 HardcopyContent

```
# Example value:  
value = enums.HardcopyContent.HCOPy  
# All values (2x):  
HCOPy | WINDows
```

3.81 HardcopyHeader

```
# Example value:  
value = enums.HardcopyHeader.ALWays  
# All values (5x):  
ALWays | GLOBal | NEVer | ONCE | SECTIon
```

3.82 HardcopyLogo

```
# Example value:  
value = enums.HardcopyLogo.ALWays  
# All values (3x):  
ALWays | GLOBal | NEVer
```

3.83 HardcopyMode

```
# Example value:
value = enums.HardcopyMode.REPort
# All values (2x):
REPort | SCReen
```

3.84 HardcopyPageSize

```
# Example value:
value = enums.HardcopyPageSize.A4
# All values (2x):
A4 | US
```

3.85 HeadersK50

```
# First value:
value = enums.HeadersK50.ALL
# Last value:
value = enums.HeadersK50.STOP
# All values (9x):
ALL | DELTa | FREQuency | IDENt | POWer | RBW | SID | START
STOP
```

3.86 HumsFileFormat

```
# Example value:
value = enums.HumsFileFormat.JSON
# All values (2x):
JSON | XML
```

3.87 IdnFormat

```
# Example value:
value = enums.IdnFormat.FSL
# All values (3x):
FSL | LEGacy | NEW
```

3.88 IfGainMode

```
# Example value:  
value = enums.IfGainMode.NORMal  
# All values (2x):  
NORMal | PULSe
```

3.89 IfGainModeDdem

```
# Example value:  
value = enums.IfGainModeDdem.AVERaging  
# All values (5x):  
AVERaging | FREeze | NORMal | TRACKing | USER
```

3.90 InOutDirection

```
# Example value:  
value = enums.InOutDirection.INPut  
# All values (2x):  
INPut | OUTPut
```

3.91 InputConnectorB

```
# Example value:  
value = enums.InputConnectorB.AIQI  
# All values (3x):  
AIQI | RF | RFProbe
```

3.92 InputConnectorC

```
# Example value:  
value = enums.InputConnectorC.RF  
# All values (2x):  
RF | RFProbe
```

3.93 InputSelect

```
# Example value:  
value = enums.InputSelect.INPut1  
# All values (2x):  
INPut1 | INPut2
```

3.94 InputSource

```
# Example value:  
value = enums.InputSource.ABBand  
# All values (7x):  
ABBand | AIQ | DIQ | FIQ | OBBand | RF | RFAiq
```

3.95 InputSourceB

```
# Example value:  
value = enums.InputSourceB.FIQ  
# All values (2x):  
FIQ | RF
```

3.96 InstrumentMode

```
# Example value:  
value = enums.InstrumentMode.MSRanalyzer  
# All values (3x):  
MSRanalyzer | RTMStandard | SANalyzer
```

3.97 IqBandwidthMode

```
# Example value:  
value = enums.IqBandwidthMode.AUTO  
# All values (3x):  
AUTO | FFT | MANual
```

3.98 IqDataFormat

```
# Example value:
value = enums.IqDataFormat.FloatComplex=FL0at32,COMpLex
# All values (4x):
FloatComplex | FloatReal | IntegerComplex | IntegerReal
```

3.99 IqDataFormatDdem

```
# First value:
value = enums.IqDataFormatDdem.FloatComplex=FL0at32,COMpLex
# Last value:
value = enums.IqDataFormatDdem.IntegerReal=INT32,REAL
# All values (10x):
FloatComplex | FloatIiQq | FloatIqIq | FloatPolar | FloatReal | IntegerComplex |   
↪ IntegerIiQq | IntegerIqIq | IntegerPolar | IntegerReal
```

3.100 IqRangeType

```
# Example value:
value = enums.IqRangeType.CAPTure
# All values (2x):
CAPTure | RRANge
```

3.101 IqResultDataFormat

```
# Example value:
value = enums.IqResultDataFormat.COMPatible
# All values (3x):
COMPatible | IQBLock | IQPair
```

3.102 IqType

```
# Example value:
value = enums.IqType.Ipart=I
# All values (3x):
Ipart | IQpart | Qpart
```

3.103 LeftRightDirection

```
# Example value:  
value = enums.LeftRightDirection.LEFT  
# All values (2x):  
LEFT | RIGHT
```

3.104 LimitState

```
# Example value:  
value = enums.LimitState.ABSolute  
# All values (4x):  
ABSolute | AND | OR | RELative
```

3.105 LoadType

```
# Example value:  
value = enums.LoadType.NEW  
# All values (2x):  
NEW | REPLace
```

3.106 LoType

```
# Example value:  
value = enums.LoType.FIXed  
# All values (2x):  
FIXed | VARiable
```

3.107 LowHigh

```
# Example value:  
value = enums.LowHigh.HIGH  
# All values (2x):  
HIGH | LOW
```


3.108 MarkerFunctionA

```
# First value:
value = enums.MarkerFunctionA.ACPower
# Last value:
value = enums.MarkerFunctionA.TPOwer
# All values (15x):
ACPower | AOBandwidth | AOBWidth | CN | CN0 | COBandwidth | COBWidth | CPOwer
GACLR | MACM | MCACpower | OBANDwidth | OBWidth | PPOwer | TPOwer
```

3.109 MarkerFunctionB

```
# Example value:
value = enums.MarkerFunctionB.ACPower
# All values (8x):
ACPower | AOBWidth | CN | CN0 | CPOwer | MCACpower | OBANDwidth | OBWidth
```

3.110 MarkerMode

```
# Example value:
value = enums.MarkerMode.DENSity
# All values (3x):
DENSity | POWer | RPOwer
```

3.111 MarkerReallmagB

```
# Example value:
value = enums.MarkerRealImagB.IMAG
# All values (2x):
IMAG | REAL
```

3.112 MeasurementStep

```
# Example value:
value = enums.MeasurementStep.NESTimate
# All values (4x):
NESTimate | SDEtection | SOVerview | SPOTstep
```

3.113 MeasurementType

```
# Example value:  
value = enums.MeasurementType.DIRected  
# All values (2x):  
DIRected | WIDE
```

3.114 MessageType

```
# Example value:  
value = enums.MessageType.REMote  
# All values (2x):  
REMote | SMSG
```

3.115 MixerIdentifier

```
# Example value:  
value = enums.MixerIdentifier.CLOCK  
# All values (2x):  
CLOCK | LO
```

3.116 MpowerDetector

```
# Example value:  
value = enums.MpowerDetector.MEAN  
# All values (2x):  
MEAN | PEAK
```

3.117 MskFormat

```
# Example value:  
value = enums.MskFormat.DIFFerential  
# All values (4x):  
DIFFerential | NORMal | TYPE1 | TYPE2
```

3.118 MspurSearchType

```
# Example value:
value = enums.MspurSearchType.DMINimum
# All values (2x):
DMINimum | PMAximum
```

3.119 MsSyncMode

```
# Example value:
value = enums.MsSyncMode.MASter
# All values (5x):
MASter | NONE | PRIMary | SECondary | SLAVe
```

3.120 NoiseFigureLimit

```
# Example value:
value = enums.NoiseFigureLimit.ENR
# All values (7x):
ENR | GAIN | NOISe | PCOLd | PHOT | TEMPerature | YFACTOR
```

3.121 NoiseFigureResult

```
# First value:
value = enums.NoiseFigureResult.CPCold
# Last value:
value = enums.NoiseFigureResult.YFACTOR
# All values (11x):
CPCold | CPHot | CYFactor | ENR | GAIN | NOISe | NUNCertainty | PCOLd
PHOT | TEMPerature | YFACTOR
```

3.122 NoiseFigureResultCustom

```
# First value:
value = enums.NoiseFigureResultCustom.CPCold
# Last value:
value = enums.NoiseFigureResultCustom.YFACTOR
# All values (10x):
CPCold | CPHot | CYFactor | ENR | GAIN | NOISe | PCOLd | PHOT
TEMPerature | YFACTOR
```

3.123 OddEven

```
# Example value:  
value = enums.OddEven.EODD  
# All values (3x):  
EODD | EVEN | ODD
```

3.124 OffState

```
# Example value:  
value = enums.OffState.OFF  
# All values (1x):  
OFF
```

3.125 OptimizationCriterion

```
# Example value:  
value = enums.OptimizationCriterion.EVMMin  
# All values (2x):  
EVMMin | RMSMin
```

3.126 OptionState

```
# Example value:  
value = enums.OptionState.OCCupy  
# All values (3x):  
OCCupy | OFF | ON
```

3.127 OutputType

```
# Example value:  
value = enums.OutputType.DEVICE  
# All values (4x):  
DEVICE | TARMed | TOFF | UDEFINED
```

3.128 PadType

```
# Example value:  
value = enums.PadType.MLPad  
# All values (2x):  
MLPad | SRESistor
```

3.129 PageMarginUnit

```
# Example value:  
value = enums.PageMarginUnit.IN  
# All values (2x):  
IN | MM
```

3.130 PageOrientation

```
# Example value:  
value = enums.PageOrientation.LANDscape  
# All values (2x):  
LANDscape | PORTrait
```

3.131 PathToCalibrate

```
# Example value:  
value = enums.PathToCalibrate.FULL  
# All values (2x):  
FULL | PARTial
```

3.132 PictureFormat

```
# First value:  
value = enums.PictureFormat.BMP  
# Last value:  
value = enums.PictureFormat.WMF  
# All values (11x):  
BMP | DOC | EWMF | GDI | JPEG | JPG | PDF | PNG  
RTF | SVG | WMF
```

3.133 PmeterFreqLink

```
# Example value:  
value = enums.PmeterFreqLink.CENter  
# All values (3x):  
CENter | MARkeR1 | OFF
```

3.134 PmRpointMode

```
# Example value:  
value = enums.PmRpointMode.MANual  
# All values (2x):  
MANual | RIGHt
```

3.135 PowerMeasFunction

```
# Example value:  
value = enums.PowerMeasFunction.ACPower  
# All values (7x):  
ACPower | CN | CN0 | CPower | MCACpower | OBANdwidth | OBWidth
```

3.136 PowerMeterUnit

```
# Example value:  
value = enums.PowerMeterUnit.DBM  
# All values (3x):  
DBM | W | WATT
```

3.137 PowerSource

```
# Example value:  
value = enums.PowerSource.VSUPply  
# All values (2x):  
VSUPply | VTUNe
```

3.138 PowerUnitB

```
# First value:
value = enums.PowerUnitB.A
# Last value:
value = enums.PowerUnitB.WATT
# All values (26x):
A | AMPere | DB | DBM | DBM_hz | DBM_mhz | DBMV | DBMV_mhz
DBPT | DBPT_mhz | DBPW | DBPW_mhz | DBUA | DBUa_m | DBUa_mhz | DBUa_mmhz
DBUV | DBUV_m | DBUV_mhz | DBUV_mmhz | PCT | UNITless | V | VOLT
W | WATT
```

3.139 PowerUnitC

```
# First value:
value = enums.PowerUnitC.A
# Last value:
value = enums.PowerUnitC.WATT
# All values (28x):
A | AMPere | DB | DBM | DBM_hz | DBM_mhz | DBMV | DBMV_mhz
DBPT | DBPT_mhz | DBPW | DBPW_mhz | DBUA | DBUa_m | DBUa_mhz | DBUa_mmhz
DBUV | DBUV_m | DBUV_mhz | DBUV_mmhz | DEG | HZ | PCT | RAD
S | UNITless | VOLT | WATT
```

3.140 PowerUnitDdem

```
# Example value:
value = enums.PowerUnitDdem.DBM
# All values (3x):
DBM | DBMV | DBUV
```

3.141 PreampOption

```
# Example value:
value = enums.PreampOption.B23
# All values (2x):
B23 | B24
```

3.142 PresetCompatible

```
# Example value:  
value = enums.PresetCompatible.MREceiver  
# All values (8x):  
MREceiver | MSRA | OFF | PNOise | REceiver | RTMS | SANalyzer | VNA
```

3.143 Probability

```
# Example value:  
value = enums.Probability.P0_01  
# All values (4x):  
P0_01 | P0_1 | P1 | P10
```

3.144 ProbeMode

```
# Example value:  
value = enums.ProbeMode.CM  
# All values (4x):  
CM | DM | NM | PM
```

3.145 ProbeSetupMode

```
# Example value:  
value = enums.ProbeSetupMode.NOAction  
# All values (2x):  
NOAction | RSINgle
```

3.146 PskFormat

```
# Example value:  
value = enums.PskFormat.DIFFerential  
# All values (7x):  
DIFFerential | DPI2 | MNPi2 | N3Pi8 | NORMa1 | NPI2 | PI8D8psk
```


3.147 PwrLevelMode

```
# Example value:  
value = enums.PwrLevelMode.CURRent  
# All values (2x):  
CURRent | VOLTage
```

3.148 PwrMeasUnit

```
# Example value:  
value = enums.PwrMeasUnit.ABS  
# All values (3x):  
ABS | PHZ | PMHZ
```

3.149 QamFormat

```
# Example value:  
value = enums.QamFormat.DIFFerential  
# All values (4x):  
DIFFerential | MNPI4 | NORMal | NPI4
```

3.150 QpskFormat

```
# Example value:  
value = enums.QpskFormat.DIFFerential  
# All values (7x):  
DIFFerential | DPI4 | N3Pi4 | NORMal | NPI4 | OFFSet | SOFFset
```

3.151 RangeClass

```
# Example value:  
value = enums.RangeClass.LOCal  
# All values (3x):  
LOCAl | MEDium | WIDE
```

3.152 RangeParam

```
# First value:  
value = enums.RangeParam.ARBW  
# Last value:  
value = enums.RangeParam.TSTR  
# All values (12x):  
ARBW | LOFFset | MFRBw | NFFT | PAValue | PEXCursion | RBW | RFATtenuation  
RLEVel | SNRatio | TSTP | TSTR
```

3.153 RefChannel

```
# Example value:  
value = enums.RefChannel.LHIGhest  
# All values (3x):  
LHIGhest | MAXimum | MINimum
```

3.154 ReferenceMode

```
# Example value:  
value = enums.ReferenceMode.ABSolute  
# All values (2x):  
ABSolute | RELative
```

3.155 ReferenceSource

```
# Example value:  
value = enums.ReferenceSource.EXT  
# All values (2x):  
EXT | INT
```

3.156 ReferenceSourceA

```
# First value:  
value = enums.ReferenceSourceA.E10  
# Last value:  
value = enums.ReferenceSourceA.SYNC  
# All values (11x):  
E10 | E100 | E1000 | EAUTo | EXT1 | EXT2 | EXTernal | EXTernal1  
EXTernal2 | INTernal | SYNC
```

3.157 ReferenceSourceB

```
# Example value:
value = enums.ReferenceSourceB.EAUto
# All values (3x):
EAUto | EXtErnal | INTernal
```

3.158 ReferenceSourceD

```
# Example value:
value = enums.ReferenceSourceD.EXT2
# All values (4x):
EXT2 | EXT3 | EXtErnal | IMMEDIATE
```

3.159 Relay

```
# First value:
value = enums.Relay.AC_enable
# Last value:
value = enums.Relay.SWRF1in
# All values (84x):
AC_enable | ACDC | AMPsw_2 | AMPsw_4 | ATT10 | ATT10db | ATT1db | ATT20
ATT20db | ATT2db | ATT40 | ATT40db | ATT4db_a | ATT4db_b | ATT5 | ATTinput2
CAL | CAL_enable | EATT | EMIMatt10 | EXT_relais | HP_Bypass | HP_Bypass_2 | HP_Hp100khz
HP_Hp1khz | HP_Hp1mhz | HP_Hp200khz | HP_Hp20khz | HP_Hp50khz | HP_Hp5mhz | HP_Hp9khz |
↳ HP_Sw
IFSW | INP2matt10r1 | INP2matt10r2 | INPut2 | LFMatt10 | LNA20db_1 | LNA20db_2 | LP_
↳ Lp100khz
LP_Lp14mhz | LP_Lp1mhz | LP_Lp20khz | LP_Lp40mhz | LP_Lp500khz | LP_Lp5mhz | LP_Sw |
↳ PREamp
PREamp30mhz | PRESab_bypr1 | PRESab_bypr2 | PRESab_swir1 | PRESab_swir2 | PRESel | RFAB_
↳ SATT10
SATT20 | SATT40 | SCAL | SIGSourout | SP6T | SPAByp | SPDTinput | SPDTmwcamp
SWAMP1 | SWAMP1amp2 | SWAMP1toamp4 | SWAMP2 | SWAMP3 | SWAMP3amp4 | SWAMP4 | SWAMP5
SWAMP6 | SWAMP7 | SWF1f2in | SWF1f2out | SWF1tof4out | SWF3f4out | SWF3in | SWF4in
SWF5in | SWF6f7in | SWFE | SWRF1in
```

3.160 ResultDevReference

```
# Example value:
value = enums.ResultDevReference.CENTer
# All values (4x):
CENTer | EDGE | FMSettling | PMSettling
```

3.161 ResultTypeB

```
# Example value:  
value = enums.ResultTypeB.ALL  
# All values (4x):  
ALL | CFACtor | MEAN | PEAK
```

3.162 ResultTypeC

```
# Example value:  
value = enums.ResultTypeC.AVERage  
# All values (2x):  
AVERage | IMMEDIATE
```

3.163 ResultTypeD

```
# Example value:  
value = enums.ResultTypeD.TOTAL  
# All values (1x):  
TOTAL
```

3.164 ResultTypeStat

```
# First value:  
value = enums.ResultTypeStat.AVG  
# Last value:  
value = enums.ResultTypeStat.TPEak  
# All values (9x):  
AVG | PAVG | PCTL | PEAK | PPCTL | PSDev | RPEak | SDEV  
TPEak
```

3.165 RoscillatorFreqMode

```
# Example value:  
value = enums.RoscillatorFreqMode.E10  
# All values (3x):  
E10 | E100 | VARIABLE
```

3.166 RoscillatorRefOut

```
# Example value:  
value = enums.RoscillatorRefOut.EXternal1  
# All values (7x):  
EXTERNAL1 | EXTERNAL2 | 010 | 0100 | 01000 | 08000 | OFF
```

3.167 ScaleYaxisUnit

```
# Example value:  
value = enums.ScaleYaxisUnit.ABS  
# All values (2x):  
ABS | PCT
```

3.168 ScalingMode

```
# Example value:  
value = enums.ScalingMode.LINEAR  
# All values (2x):  
LINEAR | LOGARITHMIC
```

3.169 SearchArea

```
# Example value:  
value = enums.SearchArea.MEMORY  
# All values (2x):  
MEMORY | VISIBLE
```

3.170 SearchRange

```
# Example value:  
value = enums.SearchRange.GMAXIMUM  
# All values (2x):  
GMAXIMUM | RMAXIMUM
```

3.171 SelectAll

```
# Example value:  
value = enums.SelectAll.ALL  
# All values (1x):  
ALL
```

3.172 SelectionRangeB

```
# Example value:  
value = enums.SelectionRangeB.ALL  
# All values (2x):  
ALL | CURRent
```

3.173 SelectionScope

```
# Example value:  
value = enums.SelectionScope.ALL  
# All values (2x):  
ALL | SINGLE
```

3.174 Separator

```
# Example value:  
value = enums.Separator.COMMa  
# All values (2x):  
COMMa | POINT
```

3.175 SequencerMode

```
# Example value:  
value = enums.SequencerMode.CDEFined  
# All values (4x):  
CDEFined | CONTInous | CONTInuous | SINGLE
```

3.176 ServiceBandwidth

```
# Example value:  
value = enums.ServiceBandwidth.BROadband  
# All values (2x):  
BROadband | NARRowband
```

3.177 ServiceState

```
# Example value:  
value = enums.ServiceState.DEViations  
# All values (4x):  
DEViations | NAN | OK | REQired
```

3.178 SidebandPos

```
# Example value:  
value = enums.SidebandPos.INVerse  
# All values (2x):  
INVerse | NORMAl
```

3.179 SignalLevel

```
# Example value:  
value = enums.SignalLevel.HIGH  
# All values (3x):  
HIGH | LOW | OFF
```

3.180 SignalType

```
# Example value:  
value = enums.SignalType.AC  
# All values (3x):  
AC | DC | DCZero
```

3.181 SingleValue

```
# First value:  
value = enums.SingleValue.ALL  
# Last value:  
value = enums.SingleValue.RHO  
# All values (13x):  
ALL | CFER | EVMP | EVMR | FDER | FEP | FERM | IQOF  
MEP | MERM | PEP | PERM | RHO
```

3.182 Size

```
# Example value:  
value = enums.Size.LARGe  
# All values (2x):  
LARGe | SMALL
```

3.183 SlopeType

```
# Example value:  
value = enums.SlopeType.NEGative  
# All values (2x):  
NEGative | POSitive
```

3.184 SnmpVersion

```
# Example value:  
value = enums.SnmpVersion.DEFaultt  
# All values (5x):  
DEFaultt | OFF | V12 | V123 | V3
```

3.185 Source

```
# First value:  
value = enums.Source.AM  
# Last value:  
value = enums.Source.VIDeo  
# All values (13x):  
AM | FM | FOCus | HVIDeo | IF | IF2 | IQ | OFF  
OUT1 | OUT2 | PM | SSB | VIDeo
```


3.186 SourceInt

```
# Example value:  
value = enums.SourceInt.EXternal  
# All values (2x):  
EXTERNAL | INTERNAL
```

3.187 SpacingY

```
# Example value:  
value = enums.SpacingY.LDB  
# All values (4x):  
LDB | LINEar | LOGarithmic | PERCent
```

3.188 SpanSetting

```
# Example value:  
value = enums.SpanSetting.FREQUENCY  
# All values (2x):  
FREQUENCY | TIME
```

3.189 State

```
# Example value:  
value = enums.State.ALL  
# All values (4x):  
ALL | AUTO | OFF | ON
```

3.190 StatisticMode

```
# Example value:  
value = enums.StatisticMode.INFINITE  
# All values (2x):  
INFINITE | SONLY
```

3.191 StatisticType

```
# Example value:  
value = enums.StatisticType.ALL  
# All values (2x):  
ALL | SElected
```

3.192 Stepsize

```
# Example value:  
value = enums.Stepsize.POINTs  
# All values (2x):  
POINTs | STANdard
```

3.193 StoreType

```
# Example value:  
value = enums.StoreType.CHANnel  
# All values (2x):  
CHANnel | INSTRument
```

3.194 SubBlockGaps

```
# Example value:  
value = enums.SubBlockGaps.AB  
# All values (7x):  
AB | BC | CD | DE | EF | FG | GH
```

3.195 SummaryMode

```
# Example value:  
value = enums.SummaryMode.AVERage  
# All values (2x):  
AVERage | SINGLe
```

3.196 SweepModeC

```
# Example value:  
value = enums.SweepModeC.AUTO  
# All values (3x):  
AUTO | ESpectrum | LIST
```

3.197 SweepOptimize

```
# Example value:  
value = enums.SweepOptimize.AUTO  
# All values (4x):  
AUTO | DYNamic | SPEed | TRANsient
```

3.198 SweepType

```
# Example value:  
value = enums.SweepType.AUTO  
# All values (3x):  
AUTO | FFT | SWEep
```

3.199 SymbolSelection

```
# Example value:  
value = enums.SymbolSelection.ALL  
# All values (3x):  
ALL | DATA | PATtern
```

3.200 Synchronization

```
# Example value:  
value = enums.Synchronization.ALL  
# All values (2x):  
ALL | NONE
```

3.201 SyncMode

```
# Example value:
value = enums.SyncMode.MEAS
# All values (2x):
MEAS | SYNC
```

3.202 SystemStatus

```
# Example value:
value = enums.SystemStatus.ERR
# All values (3x):
ERR | OK | WARN
```

3.203 TechnologyStandardA

```
# First value:
value = enums.TechnologyStandardA.GSM
# Last value:
value = enums.TechnologyStandardA.WCDMa
# All values (28x):
GSM | LTE_1_40 | LTE_10_00 | LTE_15_00 | LTE_20_00 | LTE_3_00 | LTE_5_00 | NR5G_fr1_10
NR5G_fr1_100 | NR5G_fr1_15 | NR5G_fr1_20 | NR5G_fr1_25 | NR5G_fr1_30 | NR5G_fr1_35 | ↵
↪NR5G_fr1_40 | NR5G_fr1_45
NR5G_fr1_5 | NR5G_fr1_50 | NR5G_fr1_60 | NR5G_fr1_70 | NR5G_fr1_80 | NR5G_fr1_90 | NR5G_
↪fr2_100 | NR5G_fr2_200
NR5G_fr2_400 | NR5G_fr2_50 | USER | WCDMa
```

3.204 TechnologyStandardB

```
# First value:
value = enums.TechnologyStandardB.AWLan
# Last value:
value = enums.TechnologyStandardB.WIMax
# All values (60x):
AWLan | BWLan | CDPD | D2CDma | EUTRa | F19Cdma | F1D100nr5g | F1D20nr5g
F1U100nr5g | F1U20nr5g | F2D100nr5g | F2D200nr5g | F2U100nr5g | F2U200nr5g | F8CDma | ↵
↪FIS95a
FIS95c0 | FIS95c1 | FJ008 | FTCDma | FW3Gppcdma | FWCDma | GSM | L03R
L03S | L05R | L05S | L10R | L10S | L14R | L14S | L15R
L15S | L20R | L20S | M2CDma | MSR | NONE | NR5G | NR5Glte
PAPCo25 | PDC | PHS | R19Cdma | R8CDma | REUTra | RFID14443 | RIS95a
RIS95c0 | RIS95c1 | RJ008 | RTCDma | RW3Gppcdma | RWCDma | S2CDma | TCDMa
TETRa | USER | WIBRo | WIMax
```

3.205 TechnologyStandardDdem

```
# Example value:  
value = enums.TechnologyStandardDdem.DECT  
# All values (3x):  
DECT | EDGE | GSM
```

3.206 Temperature

```
# Example value:  
value = enums.Temperature.COLD  
# All values (2x):  
COLD | HOT
```

3.207 TimeFormat

```
# Example value:  
value = enums.TimeFormat.DE  
# All values (3x):  
DE | ISO | US
```

3.208 TimeLimitUnit

```
# Example value:  
value = enums.TimeLimitUnit.S  
# All values (2x):  
S | SYM
```

3.209 TimeOrder

```
# Example value:  
value = enums.TimeOrder.AFTer  
# All values (2x):  
AFTer | BEFore
```

3.210 TouchscreenState

```
# Example value:  
value = enums.TouchscreenState.FRAME  
# All values (3x):  
FRAME | OFF | ON
```

3.211 TraceFormat

```
# First value:  
value = enums.TraceFormat.BERate  
# Last value:  
value = enums.TraceFormat.UPHase  
# All values (22x):  
BERate | BINary | COMP | CONF | CONS | COVF | DECimal | FEYE  
FREQuency | GDELay | HEXadecimal | IEYE | MAGNitude | MOVerview | NONE | OCTal  
PHASe | QEYE | RCONstell | RIMag | RSUMmary | UPHase
```

3.212 TraceModeA

```
# Example value:  
value = enums.TraceModeA.AVERage  
# All values (6x):  
AVERage | MAXHold | MINHold | OFF | VIEW | WRITe
```

3.213 TraceModeB

```
# Example value:  
value = enums.TraceModeB.AVERage  
# All values (4x):  
AVERage | MAXHold | MINHold | WRITe
```

3.214 TraceModeC

```
# Example value:  
value = enums.TraceModeC.AVERage  
# All values (6x):  
AVERage | BLANK | MAXHold | MINHold | VIEW | WRITe
```

3.215 TraceModeD

```
# Example value:
value = enums.TraceModeD.MAXHold
# All values (2x):
MAXHold | WRITe
```

3.216 TraceModeE

```
# Example value:
value = enums.TraceModeE.AVERage
# All values (3x):
AVERage | MAXHold | WRITe
```

3.217 TraceModeF

```
# Example value:
value = enums.TraceModeF.AVERage
# All values (7x):
AVERage | BLANK | DENSity | MAXHold | MINHold | VIEW | WRITe
```

3.218 TraceModeH

```
# Example value:
value = enums.TraceModeH.BLANK
# All values (3x):
BLANK | VIEW | WRITe
```

3.219 TraceNumber

```
# First value:
value = enums.TraceNumber.BTOBits
# Last value:
value = enums.TraceNumber.TRACe6
# All values (15x):
BTOBits | BTOFm | ESPLine | HMAXhold | LIST | PSpectrum | SGRam | SPEctrogram
SPURious | TRACe1 | TRACe2 | TRACe3 | TRACe4 | TRACe5 | TRACe6
```

3.220 TracePresetType

```
# Example value:
value = enums.TracePresetType.ALL
# All values (3x):
ALL | MAM | MCM
```

3.221 TraceReference

```
# Example value:
value = enums.TraceReference.BURSt
# All values (3x):
BURSt | PATtern | TRIGger
```

3.222 TraceRefType

```
# Example value:
value = enums.TraceRefType.ERROR
# All values (4x):
ERRor | MEAS | REF | TCAP
```

3.223 TraceSymbols

```
# Example value:
value = enums.TraceSymbols.BARS
# All values (4x):
BARS | DOTS | OFF | ON
```

3.224 TraceTypeDdem

```
# First value:
value = enums.TraceTypeDdem.MSTRace
# Last value:
value = enums.TraceTypeDdem.TRACe6
# All values (15x):
MSTRace | PSTRace | STRace | TRACe1 | TRACe1i | TRACe1r | TRACe2 | TRACe2i
TRACe2r | TRACe3 | TRACe3i | TRACe3r | TRACe4 | TRACe5 | TRACe6
```


3.225 TraceTypeK30

```
# Example value:
value = enums.TraceTypeK30.TRACe1
# All values (4x):
TRACe1 | TRACe2 | TRACe3 | TRACe4
```

3.226 TraceTypeK60

```
# Example value:
value = enums.TraceTypeK60.SGRam
# All values (8x):
SGRam | SPECTrogram | TRACe1 | TRACe2 | TRACe3 | TRACe4 | TRACe5 | TRACe6
```

3.227 TraceTypeList

```
# Example value:
value = enums.TraceTypeList.LIST
# All values (7x):
LIST | TRACe1 | TRACe2 | TRACe3 | TRACe4 | TRACe5 | TRACe6
```

3.228 TraceTypeNumeric

```
# Example value:
value = enums.TraceTypeNumeric.TRACe1
# All values (6x):
TRACe1 | TRACe2 | TRACe3 | TRACe4 | TRACe5 | TRACe6
```

3.229 TriggerOutType

```
# Example value:
value = enums.TriggerOutType.DEVICE
# All values (3x):
DEVICE | TARMed | UDEfined
```

3.230 TriggerPort

```
# Example value:
value = enums.TriggerPort.EXT1
# All values (4x):
EXT1 | EXT2 | HOST | OFF
```

3.231 TriggerSeqSource

```
# First value:
value = enums.TriggerSeqSource.ACVideo
# Last value:
value = enums.TriggerSeqSource.VIDeo
# All values (40x):
ACVideo | AF | AM | AMRelative | BBPower | EXT2 | EXT3 | EXT4
EXTErnal | FM | GP0 | GP1 | GP2 | GP3 | GP4 | GP5
IFPower | IMMEDIATE | INTernal | IQPower | LXI | MAGNitude | MAIT | MANual
MASK | PM | PMTRigger | PSEnsor | RFPower | SLEFt | SMONo | SMPX
SPILot | SRDS | SRIGHt | SSTereo | TDTRigger | TIME | TV | VIDeo
```

3.232 TriggerSourceB

```
# First value:
value = enums.TriggerSourceB.ACVideo
# Last value:
value = enums.TriggerSourceB.TV
# All values (30x):
ACVideo | AF | AM | AMRelative | BBPower | EXT2 | EXT3 | EXT4
EXTErnal | FM | GP0 | GP1 | GP2 | GP3 | GP4 | GP5
IFPower | IMMEDIATE | IQPower | MAGNitude | PM | RFPower | SLEFt | SMONo
SMPX | SPILot | SRDS | SRIGHt | SSTereo | TV
```

3.233 TriggerSourceD

```
# First value:
value = enums.TriggerSourceD.EXT2
# Last value:
value = enums.TriggerSourceD.VIDeo
# All values (10x):
EXT2 | EXT3 | EXT4 | EXTErnal | IFPower | IMMEDIATE | IQPower | PSEnsor
RFPower | VIDeo
```

3.234 TriggerSourceK

```
# First value:
value = enums.TriggerSourceK.EXT2
# Last value:
value = enums.TriggerSourceK.TIME
# All values (9x):
EXT2 | EXT3 | EXTERNAL | IFPower | IMMEDIATE | IQPower | MANUAL | RFPower
TIME
```

3.235 TriggerSourceL

```
# Example value:
value = enums.TriggerSourceL.EXT2
# All values (8x):
EXT2 | EXT3 | EXTERNAL | IFPower | IMMEDIATE | IQPower | RFPower | TIME
```

3.236 TriggerSourceListPower

```
# First value:
value = enums.TriggerSourceListPower.EXT2
# Last value:
value = enums.TriggerSourceListPower.VIDEO
# All values (10x):
EXT2 | EXT3 | EXT4 | EXTERNAL | IFPower | IMMEDIATE | LINE | LXI
RFPower | VIDEO
```

3.237 TriggerSourceMpower

```
# Example value:
value = enums.TriggerSourceMpower.EXT2
# All values (7x):
EXT2 | EXT3 | EXT4 | EXTERNAL | IFPower | RFPower | VIDEO
```

3.238 TuningRange

```
# Example value:
value = enums.TuningRange.SMALL
# All values (2x):
SMALL | WIDE
```

3.239 UnitMode

```
# Example value:  
value = enums.UnitMode.DB  
# All values (2x):  
DB | PCT
```

3.240 UpDownDirection

```
# Example value:  
value = enums.UpDownDirection.DOWN  
# All values (2x):  
DOWN | UP
```

3.241 UserLevel

```
# Example value:  
value = enums.UserLevel.AUTH  
# All values (3x):  
AUTH | NOAuth | PRIV
```

3.242 WindowDirection

```
# Example value:  
value = enums.WindowDirection.ABOVe  
# All values (4x):  
ABOVe | BELow | LEFT | RIGHT
```

3.243 WindowDirReplace

```
# Example value:  
value = enums.WindowDirReplace.ABOVe  
# All values (5x):  
ABOVe | BELow | LEFT | REPLace | RIGHT
```

3.244 WindowName

```
# Example value:
value = enums.WindowName.FOCus
# All values (1x):
FOCUS
```

3.245 WindowTypeBase

```
# Example value:
value = enums.WindowTypeBase.Diagram=DIAGRAM
# All values (5x):
Diagram | MarkePeakList | MarkeTable | ResultSummary | Spectrogram
```

3.246 WindowTypeIq

```
# First value:
value = enums.WindowTypeIq.Diagram=DIAGRAM
# Last value:
value = enums.WindowTypeIq.Spectrum=FREQ
# All values (9x):
Diagram | IqImagReal | IqVector | Magnitude | MarkePeakList | MarkeTable | ResultSummary |
↳ | Spectrogram
Spectrum
```

3.247 WindowTypeK30

```
# First value:
value = enums.WindowTypeK30.CalPowerCold=CPCold
# Last value:
value = enums.WindowTypeK30.YfactorResult=YFACTOR
# All values (12x):
CalPowerCold | CalPowerHot | CalYfactor | EnrMeasured | GainResult | MarkerTable |
↳ NoiseFigureResult | NoiseTemperatureResult
PowerColdResult | PowerHotResult | ResultTable | YfactorResult
```

3.248 WindowTypeK40

```
# First value:
value = enums.WindowTypeK40.FrequencyDrift=FDRift
# Last value:
value = enums.WindowTypeK40.SweepResultList=SREsults
# All values (9x):
FrequencyDrift | MarkerTable | PhaseNoiseDiagram | ResidualNoiseTable | SpectrumMonitor |
↳ SpotNoiseTable | SpurList | StabilityFreqLevel
SweepResultList
```

3.249 WindowTypeK50

```
# Example value:
value = enums.WindowTypeK50.MarkerTable=MTABLE
# All values (5x):
MarkerTable | NoiseFloorEstimate | SpectralOverview | SpurDetectionSpectrum |
↳ SpurDetectionTable
```

3.250 WindowTypeK60

```
# First value:
value = enums.WindowTypeK60.ChirpRateTimeDomain=CRTime
# Last value:
value = enums.WindowTypeK60.StatisticsTable=STABLE
# All values (15x):
ChirpRateTimeDomain | FmTimeDomain | FreqDeviationTimeDomain | IqTimeDomain |
↳ MarkerTable | ParameterDistribution | ParameterTrend | PhaseDeviationTimeDomain
PmTimeDomain | PMTimeDomainWrapped | ResultsTable | RfPowerTimeDomain | RfSpectrum |
↳ Spectrogram | StatisticsTable
```

3.251 WindowTypeK7

```
# First value:
value = enums.WindowTypeK7.AmSpectrum='XTIM:AM:RELative:AFSpectrum'
# Last value:
value = enums.WindowTypeK7.RfTimeDomain='XTIM:AM'
# All values (11x):
AmSpectrum | AmTimeDomain | FmSpectrum | FmTimeDomain | MarkerPeakList | MarkerTable |
↳ PmSpectrum | PmTimeDomain
ResultSummary | RfSpectrum | RfTimeDomain
```

3.252 WindowTypeK70

```
# First value:
value = enums.WindowTypeK70.CaptureBufferMagnAbs=CBuffer
# Last value:
value = enums.WindowTypeK70.SymbolsHexa=SYMB
# All values (9x):
CaptureBufferMagnAbs | Equalizer | ErrorVectorMagnitude | MeasMagnRel |
↳ModulationAccuracy | ModulationErrors | MultiSource | RefMagnRel
SymbolsHexa
```

3.253 Xdistribution

```
# Example value:
value = enums.Xdistribution.BINCentered
# All values (2x):
BINCentered | STARTstop
```

3.254 XyDirection

```
# Example value:
value = enums.XyDirection.HORizontal
# All values (2x):
HORizontal | VERTical
```


REPCAPS

4.1 Channel

```
# First value:
value = repcap.Channel.Ch1
# Range:
Ch1 .. Ch64
# All values (64x):
Ch1 | Ch2 | Ch3 | Ch4 | Ch5 | Ch6 | Ch7 | Ch8
Ch9 | Ch10 | Ch11 | Ch12 | Ch13 | Ch14 | Ch15 | Ch16
Ch17 | Ch18 | Ch19 | Ch20 | Ch21 | Ch22 | Ch23 | Ch24
Ch25 | Ch26 | Ch27 | Ch28 | Ch29 | Ch30 | Ch31 | Ch32
Ch33 | Ch34 | Ch35 | Ch36 | Ch37 | Ch38 | Ch39 | Ch40
Ch41 | Ch42 | Ch43 | Ch44 | Ch45 | Ch46 | Ch47 | Ch48
Ch49 | Ch50 | Ch51 | Ch52 | Ch53 | Ch54 | Ch55 | Ch56
Ch57 | Ch58 | Ch59 | Ch60 | Ch61 | Ch62 | Ch63 | Ch64
```

4.2 Colors

```
# First value:
value = repcap.Colors.Ix1
# Values (4x):
Ix1 | Ix2 | Ix3 | Ix4
```

4.3 Component

```
# First value:
value = repcap.Component.Ix1
# Range:
Ix1 .. Ix32
# All values (32x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15 | Ix16
Ix17 | Ix18 | Ix19 | Ix20 | Ix21 | Ix22 | Ix23 | Ix24
Ix25 | Ix26 | Ix27 | Ix28 | Ix29 | Ix30 | Ix31 | Ix32
```

4.4 CornerFrequency

```
# First value:  
value = repcap.CornerFrequency.Nr1  
# Range:  
Nr1 .. Nr5  
# All values (5x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5
```

4.5 DeltaMarker

```
# First value:  
value = repcap.DeltaMarker.Nr1  
# Range:  
Nr1 .. Nr32  
# All values (32x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8  
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16  
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24  
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.6 DisplayLine

```
# First value:  
value = repcap.DisplayLine.Nr1  
# Values (2x):  
Nr1 | Nr2
```

4.7 ExternalGen

```
# First value:  
value = repcap.ExternalGen.Nr1  
# Range:  
Nr1 .. Nr8  
# All values (8x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.8 ExternalPort

```
# First value:
value = repcap.ExternalPort.Nr1
# Values (3x):
Nr1 | Nr2 | Nr3
```

4.9 ExternalRosc

```
# First value:
value = repcap.ExternalRosc.Nr1
# Values (2x):
Nr1 | Nr2
```

4.10 FileList

```
# First value:
value = repcap.FileList.Nr1
# Range:
Nr1 .. Nr64
# All values (64x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
```

4.11 FilterPy

```
# First value:
value = repcap.FilterPy.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.12 FreqLine

```
# First value:  
value = repcap.FreqLine.Nr1  
# Values (4x):  
Nr1 | Nr2 | Nr3 | Nr4
```

4.13 FreqOffset

```
# First value:  
value = repcap.FreqOffset.Nr1  
# Range:  
Nr1 .. Nr8  
# All values (8x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.14 GapChannel

```
# First value:  
value = repcap.GapChannel.Nr1  
# Values (2x):  
Nr1 | Nr2
```

4.15 GateRange

```
# First value:  
value = repcap.GateRange.Nr1  
# Range:  
Nr1 .. Nr64  
# All values (64x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8  
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16  
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24  
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32  
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40  
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48  
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56  
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
```

4.16 Generator

```
# First value:
value = repcap.Generator.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.17 InputIx

```
# First value:
value = repcap.InputIx.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.18 Item

```
# First value:
value = repcap.Item.Ix1
# Range:
Ix1 .. Ix64
# All values (64x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15 | Ix16
Ix17 | Ix18 | Ix19 | Ix20 | Ix21 | Ix22 | Ix23 | Ix24
Ix25 | Ix26 | Ix27 | Ix28 | Ix29 | Ix30 | Ix31 | Ix32
Ix33 | Ix34 | Ix35 | Ix36 | Ix37 | Ix38 | Ix39 | Ix40
Ix41 | Ix42 | Ix43 | Ix44 | Ix45 | Ix46 | Ix47 | Ix48
Ix49 | Ix50 | Ix51 | Ix52 | Ix53 | Ix54 | Ix55 | Ix56
Ix57 | Ix58 | Ix59 | Ix60 | Ix61 | Ix62 | Ix63 | Ix64
```

4.19 LimitIx

```
# First value:
value = repcap.LimitIx.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.20 Line

```
# First value:
value = repcap.Line.Ix1
# Range:
Ix1 .. Ix8
# All values (8x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
```

4.21 Marker

```
# First value:
value = repcap.Marker.Nr1
# Range:
Nr1 .. Nr16
# All values (16x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
```

4.22 MarkerDestination

```
# First value:
value = repcap.MarkerDestination.Nr1
# Range:
Nr1 .. Nr16
# All values (16x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
```

4.23 OutputConnector

```
# First value:
value = repcap.OutputConnector.Nr1
# Values (4x):
Nr1 | Nr2 | Nr3 | Nr4
```

4.24 Port

```
# First value:
value = repcap.Port.Nr1
# Values (2x):
Nr1 | Nr2
```

4.25 PowerClass

```
# First value:
value = repcap.PowerClass.Nr1
# Range:
Nr1 .. Nr30
# All values (30x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30
```

4.26 PowerMeter

```
# First value:
value = repcap.PowerMeter.Nr1
# Range:
Nr1 .. Nr16
# All values (16x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
```

4.27 Probe

```
# First value:
value = repcap.Probe.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.28 RangePy

```
# First value:
value = repcap.RangePy.Ix1
# Range:
Ix1 .. Ix64
# All values (64x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15 | Ix16
Ix17 | Ix18 | Ix19 | Ix20 | Ix21 | Ix22 | Ix23 | Ix24
Ix25 | Ix26 | Ix27 | Ix28 | Ix29 | Ix30 | Ix31 | Ix32
Ix33 | Ix34 | Ix35 | Ix36 | Ix37 | Ix38 | Ix39 | Ix40
Ix41 | Ix42 | Ix43 | Ix44 | Ix45 | Ix46 | Ix47 | Ix48
Ix49 | Ix50 | Ix51 | Ix52 | Ix53 | Ix54 | Ix55 | Ix56
Ix57 | Ix58 | Ix59 | Ix60 | Ix61 | Ix62 | Ix63 | Ix64
```

4.29 RefMeasurement

```
# First value:
value = repcap.RefMeasurement.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.30 Source

```
# First value:
value = repcap.Source.Nr1
# Values (2x):
Nr1 | Nr2
```


4.31 SPortPair

```
# First value:
value = repcap.SPortPair.Ix1
# Values (4x):
Ix1 | Ix2 | Ix3 | Ix4
```

4.32 Status

```
# First value:
value = repcap.Status.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.33 Step

```
# First value:
value = repcap.Step.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.34 Store

```
# First value:
value = repcap.Store.Pos1
# Range:
Pos1 .. Pos32
# All values (32x):
Pos1 | Pos2 | Pos3 | Pos4 | Pos5 | Pos6 | Pos7 | Pos8
Pos9 | Pos10 | Pos11 | Pos12 | Pos13 | Pos14 | Pos15 | Pos16
Pos17 | Pos18 | Pos19 | Pos20 | Pos21 | Pos22 | Pos23 | Pos24
Pos25 | Pos26 | Pos27 | Pos28 | Pos29 | Pos30 | Pos31 | Pos32
```

4.35 SubBlock

```
# First value:
value = repcap.SubBlock.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.36 SubWindow

```
# First value:
value = repcap.SubWindow.Nr1
# Range:
Nr1 .. Nr32
# All values (32x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

4.37 TouchStone

```
# First value:
value = repcap.TouchStone.Ix1
# Range:
Ix1 .. Ix32
# All values (32x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15 | Ix16
Ix17 | Ix18 | Ix19 | Ix20 | Ix21 | Ix22 | Ix23 | Ix24
Ix25 | Ix26 | Ix27 | Ix28 | Ix29 | Ix30 | Ix31 | Ix32
```

4.38 Trace

```
# First value:
value = repcap.Trace.Tr1
# Range:
Tr1 .. Tr16
# All values (16x):
Tr1 | Tr2 | Tr3 | Tr4 | Tr5 | Tr6 | Tr7 | Tr8
Tr9 | Tr10 | Tr11 | Tr12 | Tr13 | Tr14 | Tr15 | Tr16
```

4.39 TriggerPort

```
# First value:
value = repcap.TriggerPort.Nr1
# Range:
Nr1 .. Nr8
# All values (8x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
```

4.40 UpperAltChannel

```
# First value:
value = repcap.UpperAltChannel.Nr1
# Range:
Nr1 .. Nr63
# All values (63x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63
```

4.41 UserRange

```
# First value:
value = repcap.UserRange.Nr1
# Range:
Nr1 .. Nr16
# All values (16x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
```

4.42 Window

```
# First value:
value = repcap.Window.Nr1
# Range:
Nr1 .. Nr16
# All values (16x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
```

4.43 ZoomWindow

```
# First value:  
value = repcap.ZoomWindow.Nr1  
# Range:  
Nr1 .. Nr32  
# All values (32x):  
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8  
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16  
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24  
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
```

EXAMPLES

For more examples, visit our [Rohde & Schwarz Github repository](#).

```
"""Getting started - how to work with RsFswp Python SCPI package.
This example performs basic RF settings and measurements on an FSW instrument.
It shows the RsFswp calls and their corresponding SCPI commands.
Notice that the python RsFswp interfaces track the SCPI commands syntax."""

from RsFswp import *

# A good practice is to check for the installed version
RsFswp.assert_minimum_version('2.0.0')

# Open the session
fswp = RsFswp('TCPIP::localhost::HISLIP', reset=True)
# Greetings, stranger...
print(f'Hello, I am: {fswp.utilities.idn_string}')

# Print commands to the console with the logger
fswp.utilities.logger.mode = LoggingMode.On
fswp.utilities.logger.log_to_console = True

# Driver's instrument status checking ( SYST:ERR? ) after each command (default value is_
↳ true):
fswp.utilities.instrument_status_checking = True

#  SYSTem:DISPlay:UPDate ON
fswp.system.display.update.set(True)

#  INITiate:CONTinuous OFF
fswp.initiate.continuous.set(False)
print(f'Always work in single-sweep mode.')

#  SENSE:FREQuency:STARt 1000000000
fswp.sense.frequency.start.set(100E6)

#  SENSE:FREQuency:STOP 2000000000
fswp.sense.frequency.stop.set(200E6)

#  DISPlay:WINDow:TRACe:Y:SCALe:RLEVel -20.0
fswp.display.window.trace.y.scale.refLevel.set(-20.0)
```

(continues on next page)

(continued from previous page)

```

#   DISPlay1:WINDow:SUBWindow:TRACe1:MODE:MAXHold
fswp.display.window.subwindow.trace.mode.set(enums.TraceModeC.MAXHold, repcap.Window.Nr1,
↪ repcap.SubWindow.Default, repcap.Trace.Tr1)

#   DISPlay1: WINDow:SUBWindow:TRACe2:MODE MINHold
fswp.display.window.subwindow.trace.mode.set(enums.TraceModeC.MINHold, repcap.Window.Nr1,
↪ repcap.SubWindow.Default, repcap.Trace.Tr2)

#   SENSE:SWEep:POINts 10001
fswp.sense.sweep.points.set(10001)

#   INITiate:IMMediate (with timeout 3000 ms)
fswp.initiate.immediate_with_opc(3000)

#           TRACe1:DATA?
trace1 = fswp.trace.data.get(enums.TraceNumber.TRACe1)

#           TRACe2:DATA?
trace2 = fswp.trace.data.get(enums.TraceNumber.TRACe2)

#   CALCulate1:MARKer1:TRACe 1
fswp.calculate.marker.trace.set(1, repcap.Window.Nr1, repcap.Marker.Nr1)

#   CALCulate1:MARKer1:MAXimum:PEAK
fswp.calculate.marker.maximum.peak.set(repcap.Window.Nr1, repcap.Marker.Nr1)
#           CALCulate1:MARKer1:X?
m1x = fswp.calculate.marker.x.get(repcap.Window.Nr1, repcap.Marker.Nr1)

#           CALCulate1:MARKer1:Y?
m1y = fswp.calculate.marker.y.get(repcap.Window.Nr1, repcap.Marker.Nr1)

print(f'Trace 1 points: {len(trace1)}')
print(f'Trace 1 Marker 1: {m1x} Hz, {m1y} dBm')

#   CALCulate1:MARKer2:TRACe 2
fswp.calculate.marker.trace.set(2, repcap.Window.Nr1, repcap.Marker.Nr2)

#   CALCulate1:MARKer2:MINimum:PEAK
fswp.calculate.marker.minimum.peak.set(repcap.Window.Nr1, repcap.Marker.Nr2)

#           CALCulate2:MARKer2:X?
m2x = fswp.calculate.marker.x.get(repcap.Window.Nr1, repcap.Marker.Nr2)

#           CALCulate2:MARKer2:Y?
m2y = fswp.calculate.marker.y.get(repcap.Window.Nr1, repcap.Marker.Nr2)

print(f'Trace 1 points: {len(trace2)}')
print(f'Trace 1 Marker 1: {m2x} Hz, {m2y} dBm')

# Close the session
fswp.close()

```

RSFSWP API STRUCTURE

class RsFswp(*resource_name: str, id_query: bool = True, reset: bool = False, options: Optional[str] = None, direct_session: Optional[object] = None*)

2676 total commands, 24 Subgroups, 1 group commands

Initializes new RsFswp session.

Parameter options tokens examples:

- **Simulate=True** - starts the session in simulation mode. Default: **False**
- **SelectVisa=socket** - uses no VISA implementation for socket connections - you do not need any VISA-C installation
- **SelectVisa=rs** - forces usage of RohdeSchwarz Visa
- **SelectVisa=ivi** - forces usage of National Instruments Visa
- **QueryInstrumentStatus = False** - same as **driver.utilities.instrument_status_checking = False**. Default: **True**
- **WriteDelay = 20, ReadDelay = 5** - Introduces delay of 20ms before each write and 5ms before each read. Default: **0ms** for both
- **OpcWaitMode = OpcQuery** - mode for all the opc-synchronised write/reads. Other modes: **StbPolling, StbPollingSlow, StbPollingSuperSlow**. Default: **StbPolling**
- **AddTermCharToWriteBinBlock = True** - Adds one additional LF to the end of the binary data (some instruments require that). Default: **False**
- **AssureWriteWithTermChar = True** - Makes sure each command/query is terminated with termination character. Default: Interface dependent
- **TerminationCharacter = "\r"** - Sets the termination character for reading. Default: **\n** (LineFeed or LF)
- **DataChunkSize = 10E3** - Maximum size of one write/read segment. If transferred data is bigger, it is split to more segments. Default: **1E6** bytes
- **OpcTimeout = 10000** - same as **driver.utilities.opc_timeout = 10000**. Default: **30000ms**
- **VisaTimeout = 5000** - same as **driver.utilities.visa_timeout = 5000**. Default: **10000ms**
- **ViClearExeMode = Disabled** - **viClear()** execution mode. Default: **execute_on_all**
- **OpcQueryAfterWrite = True** - same as **driver.utilities.opc_query_after_write = True**. Default: **False**
- **StbInErrorCheck = False** - if true, the driver checks errors with ***STB?** If false, it uses **SYST:ERR?**. Default: **True**

- `LoggingMode = On` - Sets the logging status right from the start. Default: `Off`
- `LoggingName = 'MyDevice'` - Sets the name to represent the session in the log entries. Default: `'resource_name'`
- `LogToGlobalTarget = True` - Sets the logging target to the class-property previously set with `RsFswp.set_global_logging_target()` Default: `False`
- `LoggingToConsole = True` - Immediately starts logging to the console. Default: `False`
- `LoggingToUdp = True` - Immediately starts logging to the UDP port. Default: `False`
- `LoggingUdpPort = 49200` - UDP port to log to. Default: `49200`

Parameters

- **resource_name** – VISA resource name, e.g. `'TCPIP::192.168.2.1::INSTR'`
- **id_query** – if `True`, the instrument's model name is verified against the models supported by the driver and eventually throws an exception.
- **reset** – Resets the instrument (sends `*RST` command) and clears its status subsystem.
- **options** – string tokens alternating the driver settings.
- **direct_session** – Another driver object or `pyVisa` object to reuse the session instead of opening a new session.

`abort()` → `None`

```
# SCPI: ABORT
driver.abort()
```

This command aborts the measurement in the current channel and resets the trigger system. To prevent overlapping execution of the subsequent command before the measurement has been aborted successfully, use the `*OPC?` or `*WAI` command after method `RsFswp.#Abort CMDLINKRESOLVED` and before the next command. For details on overlapping execution see Remote control via SCPI. To abort a sequence of measurements by the Sequencer, use the `[CMDLINKRESOLVED Initiate.Sequencer.abort` command. Note on blocked remote control programs: If a sequential command cannot be completed, for example because a triggered sweep never receives a trigger, the remote control program will never finish and the remote channel to the R&S FSWP is blocked for further commands. In this case, you must interrupt processing on the remote channel first in order to abort the measurement. To do so, send a 'Device Clear' command from the control instrument to the R&S FSWP on a parallel channel to clear all currently active remote channels. Depending on the used interface and protocol, send the following commands:

- Visa: `viClear`
- GPIB: `ibclr`
- RSIB: `RSDLLibclr`

Now you can send the `[CMDLINKRESOLVED #Abort CMDLINKRESOLVED]` command on the remote channel performing the measurement.

`abort_with_opc(opc_timeout_ms: int = -1)` → `None`

```
# SCPI: ABORT
driver.abort_with_opc()
```

This command aborts the measurement in the current channel and resets the trigger system. To prevent overlapping execution of the subsequent command before the measurement has been aborted successfully, use the `*OPC?` or `*WAI` command after method `RsFswp.#Abort CMDLINKRESOLVED` and before the

next command. For details on overlapping execution see Remote control via SCPI. To abort a sequence of measurements by the Sequencer, use the [CMDLINKRESOLVED Initiate.Sequencer.abort command. Note on blocked remote control programs: If a sequential command cannot be completed, for example because a triggered sweep never receives a trigger, the remote control program will never finish and the remote channel to the R&S FSWP is blocked for further commands. In this case, you must interrupt processing on the remote channel first in order to abort the measurement. To do so, send a 'Device Clear' command from the control instrument to the R&S FSWP on a parallel channel to clear all currently active remote channels. Depending on the used interface and protocol, send the following commands:

- Visa: viClear
- GPIB: ibclr
- RSIB: RSDLLibclr

Now you can send the [CMDLINKRESOLVED #Abort CMDLINKRESOLVED] command on the remote channel performing the measurement.

Same as abort, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

static assert_minimum_version(min_version: str) → None

Asserts that the driver version fulfills the minimum required version you have entered. This way you make sure your installed driver is of the entered version or newer.

classmethod clear_global_logging_relative_timestamp() → None

Clears the global relative timestamp. After this, all the instances using the global relative timestamp continue logging with the absolute timestamps.

close() → None

Closes the active RsFswp session.

classmethod from_existing_session(session: object, options: Optional[str] = None) → RsFswp

Creates a new RsFswp object with the entered 'session' reused.

Parameters

- **session** – can be another driver or a direct pyvisa session.
- **options** – string tokens alternating the driver settings.

classmethod get_global_logging_relative_timestamp() → datetime

Returns global common relative timestamp for log entries.

classmethod get_global_logging_target()

Returns global common target stream.

get_session_handle() → object

Returns the underlying session handle.

get_total_execution_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than get_total_time(), since it does not include gaps between the communication. You can reset this counter with reset_time_statistics().

get_total_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than `get_total_time()`, since it does not include gaps between the communication. You can reset this counter with `reset_time_statistics()`.

static list_resources(*expression: str = '?*::INSTR', visa_select: Optional[str] = None*) → List[str]

Finds all the resources defined by the expression

- `'?*' - matches all the available instruments`
- `'USB::?*' - matches all the USB instruments`
- `'TCPIP::192?*' - matches all the LAN instruments with the IP address starting with 192`

Parameters

- **expression** – see the examples in the function
- **visa_select** – optional parameter selecting a specific VISA. Examples: `'@ivi'`, `'@rs'`

reset_time_statistics() → None

Resets all execution and total time counters. Affects the results of `get_total_time()` and `get_total_execution_time()`

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

classmethod set_global_logging_relative_timestamp(*timestamp: datetime*) → None

Sets global common relative timestamp for log entries. To use it, call the following: `io.utilities.logger.set_relative_timestamp_global()`

classmethod set_global_logging_relative_timestamp_now() → None

Sets global common relative timestamp for log entries to this moment. To use it, call the following: `io.utilities.logger.set_relative_timestamp_global()`.

classmethod set_global_logging_target(*target*) → None

Sets global common target stream that each instance can use. To use it, call the following: `io.utilities.logger.set_logging_target_global()`. If an instance uses global logging target, it automatically uses the global relative timestamp (if set). You can set the target to None to invalidate it.

Subgroups

6.1 Applications

class ApplicationsCls

Applications commands group definition. 1237 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.clone()
```

Subgroups

6.1.1 IqAnalyzer

class IqAnalyzerCls

IqAnalyzer commands group definition. 52 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.clone()
```

Subgroups

6.1.1.1 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.iqAnalyzer.inputPy.repcap_inputIx_get()
driver.applications.iqAnalyzer.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 18 total commands, 1 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.clone()
```

Subgroups

6.1.1.1.1 Iq

class IqCls

Iq commands group definition. 18 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.clone()
```

Subgroups

6.1.1.1.1.1 Balanced

class BalancedCls

Balanced commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.balanced.clone()
```

Subgroups

6.1.1.1.1.2 State

SCPI Commands

```
INPut<InputIx>:IQ:BAnced:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:IQ:BAnced[:STATe]
value: bool = driver.applications.iqAnalyzer.inputPy.iq.balanced.state.
↳ get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:BAnced[:STATe]
driver.applications.iqAnalyzer.inputPy.iq.balanced.state.set(state = False,
↳ inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.1.1.3 Fullscale**class FullscaleCls**

Fullscale commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.fullscale.clone()
```

Subgroups**6.1.1.1.1.4 Level****SCPI Commands**

```
INPut<InputIx>:IQ:FULLscale:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:FULLscale[:LEVel]
value: float = driver.applications.iqAnalyzer.inputPy.iq.fullscale.level.
↳get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

level: No help available

set(level: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:FULLscale[:LEVel]
driver.applications.iqAnalyzer.inputPy.iq.fullscale.level.set(level = 1.0,
↳inputIx = repcap.InputIx.Default)
```

No command help available

param level

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.1.1.5 Impedance**class ImpedanceCls**

Impedance commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.impedance.clone()
```

Subgroups**6.1.1.1.1.6 Ptype****SCPI Commands**

```
INPut<InputIx>:IQ:IMPedance:PTYPE
```

class PtypeCls

Ptype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → PadType

```
# SCPI: INPut<ip>:IQ:IMPedance:PTYPE
value: enums.PadType = driver.applications.iqAnalyzer.inputPy.iq.impedance.
↳ptype.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

pad_type: No help available

set(pad_type: PadType, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:IMPedance:PTYPE
driver.applications.iqAnalyzer.inputPy.iq.impedance.ptype.set(pad_type = enums.
↳PadType.MLPad, inputIx = repcap.InputIx.Default)
```

No command help available

param pad_type

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.1.1.7 Osc**class OscCls**

Osc commands group definition. 14 total commands, 11 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.clone()
```

Subgroups**6.1.1.1.1.8 Balanced****class BalancedCls**

Balanced commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.balanced.clone()
```

Subgroups**6.1.1.1.1.9 State****SCPI Commands**

```
INPut<InputIx>:IQ:OSC:BALanced:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:IQ:OSC:BALanced[:STATE]
value: bool = driver.applications.iqAnalyzer.inputPy.iq.osc.balanced.state.
    ↪ get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:BALanced[:STATe]
driver.applications.iqAnalyzer.inputPy.iq.osc.balanced.state.set(state = False,
↪ inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.1.1.1.10 ConState

SCPI Commands

```
INPut<InputIx>:IQ:OSC:CONState
```

class ConStateCls

ConState commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → str

```
# SCPI: INPut<ip>:IQ:OSC:CONState
value: str = driver.applications.iqAnalyzer.inputPy.iq.osc.conState.get(inputIx,
↪ = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

connection_state: No help available

6.1.1.1.1.11 Coupling

SCPI Commands

```
INPut<InputIx>:IQ:OSC:COUPling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → CouplingTypeB


```
# SCPI: INPut<ip>:IQ:OSC:COUPling
value: enums.CouplingTypeB = driver.applications.iqAnalyzer.inputPy.iq.osc.
↪coupling.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

coupling_type: No help available

set(coupling_type: CouplingTypeB, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:COUPling
driver.applications.iqAnalyzer.inputPy.iq.osc.coupling.set(coupling_type = ↪
↪enums.CouplingTypeB.AC, inputIx = repcap.InputIx.Default)
```

No command help available

param coupling_type

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.1.1.12 Fullscale

class FullscaleCls

Fullscale commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.fullscale.clone()
```

Subgroups

6.1.1.1.13 Level

SCPI Commands

```
INPut<InputIx>:IQ:OSC:FULLscale:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → float

```
# SCPI: INPut<ip>:IQ:OSC:FULLscale[:LEVel]
value: float = driver.applications.iqAnalyzer.inputPy.iq.osc.fullscale.level.
↪get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

level: No help available

set(*level: float, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:FULLscale[:LEVel]
driver.applications.iqAnalyzer.inputPy.iq.osc.fullscale.level.set(level = 1.0, ↪
↪inputIx = repcap.InputIx.Default)
```

No command help available

param level

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.1.1.14 Idn

SCPI Commands

```
INPut<InputIx>:IQ:OSC:IDN
```

class IdnCIs

Idn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → str

```
# SCPI: INPut<ip>:IQ:OSC:IDN
value: str = driver.applications.iqAnalyzer.inputPy.iq.osc.idn.get(inputIx = ↪
↪repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

identification: No help available

6.1.1.1.15 Skew

class SkewCls

Skew commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.clone()
```

Subgroups

6.1.1.1.16 Icomponent

SCPI Commands

```
INPut<InputIx>:IQ:OSC:SKEW:I
```

class IcomponentCls

Icomponent commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I
value: float = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.icomponent.
↳get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I
driver.applications.iqAnalyzer.inputPy.iq.osc.skew.icomponent.set(value = 1.0,
↳inputIx = repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.icomponent.clone()
```

Subgroups

6.1.1.1.17 Inverted

SCPI Commands

```
INPut<InputIx>:IQ:OSC:SKEW:I:INVerted
```

class InvertedCls

Inverted commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I:INVerted
value: float = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.icomponent.
↳inverted.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I:INVerted
driver.applications.iqAnalyzer.inputPy.iq.osc.skew.icomponent.inverted.
↳set(value = 1.0, inputIx = repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.1.1.18 Qcomponent

SCPI Commands

```
INPut<InputIx>:IQ:OSC:SKEW:Q
```

class QcomponentCls

Qcomponent commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q
value: float = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.qcomponent.
↳get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

value: No help available

set(*value: float, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q
driver.applications.iqAnalyzer.inputPy.iq.osc.skew.qcomponent.set(value = 1.0,
↳inputIx = repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.qcomponent.clone()
```

Subgroups

6.1.1.1.19 Inverted

SCPI Commands

```
INPut<InputIx>:IQ:OSC:SKEW:Q:INVerted
```

class InvertedCls

Inverted commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q:INVerted
value: float = driver.applications.iqAnalyzer.inputPy.iq.osc.skew.qcomponent.
↳inverted.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q:INVerted
driver.applications.iqAnalyzer.inputPy.iq.osc.skew.qcomponent.inverted.
↳set(value = 1.0, inputIx = repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.1.20 State

SCPI Commands

```
INPut<InputIx>:IQ:OSC:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:IQ:OSC[:STATe]
value: bool = driver.applications.iqAnalyzer.inputPy.iq.osc.state.get(inputIx =
↳repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC[:STATe]
driver.applications.iqAnalyzer.inputPy.iq.osc.state.set(state = False, inputIx =
↳ repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.1.1.1.21 Tcpip

SCPI Commands

```
INPut<InputIx>:IQ:OSC:TCPIP
```

class TcpipCls

Tcpip commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → str

```
# SCPI: INPut<ip>:IQ:OSC:TCPIP
value: str = driver.applications.iqAnalyzer.inputPy.iq.osc.tcpip.get(inputIx =
↳ repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

ip_address: No help available

6.1.1.1.1.22 TypePy

SCPI Commands

```
INPut<InputIx>:IQ:OSC:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → IqType

```
# SCPI: INPut<ip>:IQ:OSC:TYPE
value: enums.IqType = driver.applications.iqAnalyzer.inputPy.iq.osc.typePy.
↳ get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

type_py: No help available

set(type_py: *IqType*, inputIx=*InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:TYPE
driver.applications.iqAnalyzer.inputPy.iq.osc.typePy.set(type_py = enums.IqType.
↪ Ipart=I, inputIx = repcap.InputIx.Default)
```

No command help available

param type_py

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.1.1.23 Vdevice

SCPI Commands

```
INPut<InputIx>:IQ:OSC:VDEvice
```

class VdeviceCls

Vdevice commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=*InputIx.Default*) → bool

```
# SCPI: INPut<ip>:IQ:OSC:VDEvice
value: bool = driver.applications.iqAnalyzer.inputPy.iq.osc.vdevice.get(inputIx,
↪ repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

6.1.1.1.24 Vfirmware

SCPI Commands

```
INPut<InputIx>:IQ:OSC:VFIRmware
```

class VfirmwareCls

Vfirmware commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → bool

```
# SCPI: INPut<ip>:IQ:OSC:VFIRmware
value: bool = driver.applications.iqAnalyzer.inputPy.iq.osc.vfirmware.
↪get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

6.1.1.1.25 TypePy

SCPI Commands

```
INPut<InputIx>:IQ:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → IqType

```
# SCPI: INPut<ip>:IQ:TYPE
value: enums.IqType = driver.applications.iqAnalyzer.inputPy.iq.typePy.
↪get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

data_type: No help available

set(*data_type: IqType, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:TYPE
driver.applications.iqAnalyzer.inputPy.iq.typePy.set(data_type = enums.IqType.
↪Ipart=I, inputIx = repcap.InputIx.Default)
```

No command help available

param data_type

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.1.2 Layout

class LayoutCls

Layout commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.layout.clone()
```

Subgroups

6.1.1.2.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.layout.add.clone()
```

Subgroups

6.1.1.2.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeIq) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.iqAnalyzer.layout.add.window.get(window_name =
↳ '1', direction = enums.WindowDirection.ABOVe, window_type = enums.
↳ WindowTypeIq.Diagram=DIAGram)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method `RsFswp.Layout.Replace.Window.set` command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method `RsFswp.Layout.Catalog.Window.get_query`.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

`new_window_name`: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.1.2.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.layout.catalog.clone()
```

Subgroups

6.1.1.2.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get()` → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.iqAnalyzer.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return
result: No help available

6.1.1.2.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.layout.identify.clone()
```

Subgroups

6.1.1.2.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.iqAnalyzer.layout.identify.window.get(window_
name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name
String containing the name of a window.

return
window_index: Index number of the window.

6.1.1.2.4 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.layout.replace.clone()
```

Subgroups

6.1.1.2.4.1 Window

SCPI Commands

```
LAYout:REPLace:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeIq) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.iqAnalyzer.layout.replace.window.set(window_name = '1',
↳window_type = enums.WindowTypeIq.Diagram=DIAGram)
```

This command replaces the window type (for example from 'Diagram' to 'Result Summary') of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method RsFswp.Layout. **Add.Window.get_** command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method **RsFswp.Layout.Catalog.Window.get_** query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method **RsFswp.Layout.Add.Window.get_** for a list of available window types.

6.1.1.3 Sense

class SenseCls

Sense commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.sense.clone()
```

Subgroups

6.1.1.3.1 Iq

class IqCls

Iq commands group definition. 7 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.sense.iq.clone()
```

Subgroups

6.1.1.3.1.1 Fft

class FftCls

Fft commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.sense.iq.fft.clone()
```

Subgroups

6.1.1.3.1.2 Algorithm

SCPI Commands

```
SENSe:IQ:FFT:ALGORITHM
```

class AlgorithmCls

Algorithm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SummaryMode

```
# SCPI: [SENSe]:IQ:FFT:ALGORITHM
value: enums.SummaryMode = driver.applications.iqAnalyzer.sense.iq.fft.
    ↪ algorithm.get()
```

Defines the FFT calculation method. For more information see ‘Basics on FFT’.

```
return
    algorithm: No help available
```

set(*algorithm: SummaryMode*) → None

```
# SCPI: [SENSe]:IQ:FFT:ALGorithm
driver.applications.iqAnalyzer.sense.iq.fft.algorithm.set(algorithm = enums.
↳SummaryMode.AVERage)
```

Defines the FFT calculation method. For more information see ‘Basics on FFT’.

param algorithm

SINGLE One FFT is calculated for the entire record length; if the FFT length is larger than the record length (see [SENSe]:IQ:FFT:LENGth and method RsFswp.Applications.IqAnalyzer.Trace.Iq.Rlength.set) , zeros are appended to the captured data. AVERage Several overlapping FFTs are calculated for each record; the results are averaged to determine the final FFT result for the record. The user-defined window length and window overlap are used (see [SENSe]:IQ:FFT:WINDow:LENGth and [SENSe]:IQ:FFT:WINDow:OVERlap) .

6.1.1.3.1.3 Length

SCPI Commands

SENSe:IQ:FFT:LENGth

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:IQ:FFT:LENGth
value: int = driver.applications.iqAnalyzer.sense.iq.fft.length.get()
```

Defines the number of frequency points determined by each FFT calculation. The more points are used, the higher the resolution in the spectrum becomes, but the longer the calculation takes.

return

length: No help available

set(*length: int*) → None

```
# SCPI: [SENSe]:IQ:FFT:LENGth
driver.applications.iqAnalyzer.sense.iq.fft.length.set(length = 1)
```

Defines the number of frequency points determined by each FFT calculation. The more points are used, the higher the resolution in the spectrum becomes, but the longer the calculation takes.

param length

integer value Range: 3 to 524288

6.1.1.3.1.4 Window

class WindowCls

Window commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.sense.iq.fft.window.clone()
```

Subgroups

6.1.1.3.1.5 Length

SCPI Commands

```
SENSe:IQ:FFT:WINDow:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:IQ:FFT:WINDow:LENGth
value: int = driver.applications.iqAnalyzer.sense.iq.fft.window.length.get()
```

Defines the number of samples to be included in a single FFT window when multiple FFT windows are used.

return

length: No help available

set(length: int) → None

```
# SCPI: [SENSe]:IQ:FFT:WINDow:LENGth
driver.applications.iqAnalyzer.sense.iq.fft.window.length.set(length = 1)
```

Defines the number of samples to be included in a single FFT window when multiple FFT windows are used.

param length

integer value Range: 3 to 1001

6.1.1.3.1.6 Overlap

SCPI Commands

```
SENSe:IQ:FFT:WINDow:OVERlap
```

class OverlapCls

Overlap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:IQ:FFT:WINDow:OVERlap
value: float = driver.applications.iqAnalyzer.sense.iq.fft.window.overlap.get()
```

Defines the part of a single FFT window that is re-calculated by the next FFT calculation.

return
overlap: No help available

set(overlap: float) → None

```
# SCPI: [SENSe]:IQ:FFT:WINDow:OVERlap
driver.applications.iqAnalyzer.sense.iq.fft.window.overlap.set(overlap = 1.0)
```

Defines the part of a single FFT window that is re-calculated by the next FFT calculation.

param overlap
double value Percentage rate Range: 0 to 1

6.1.1.3.1.7 TypePy

SCPI Commands

```
SENSe:IQ:FFT:WINDow:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FftWindowType

```
# SCPI: [SENSe]:IQ:FFT:WINDow:TYPE
value: enums.FftWindowType = driver.applications.iqAnalyzer.sense.iq.fft.window.
↳ typePy.get()
```

In the I/Q Analyzer you can select one of several FFT window types.

return
type_py: No help available

set(type_py: FftWindowType) → None

```
# SCPI: [SENSe]:IQ:FFT:WINDow:TYPE
driver.applications.iqAnalyzer.sense.iq.fft.window.typePy.set(type_py = enums.
↳ FftWindowType.BLACKharris)
```

In the I/Q Analyzer you can select one of several FFT window types.

param type_py

BLACKharris Blackman-Harris FLATtop Flattop GAUSSsian Gauss RECTangular
Rectangular P5 5-Term

6.1.1.3.1.8 Impedance

SCPI Commands

SENSe:IQ:IMPedance

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:IQ:IMPedance
value: int = driver.applications.iqAnalyzer.sense.iq.impedance.get()
```

No command help available

return

impedance: No help available

set(impedance: int) → None

```
# SCPI: [SENSe]:IQ:IMPedance
driver.applications.iqAnalyzer.sense.iq.impedance.set(impedance = 1)
```

No command help available

param impedance

No help available

6.1.1.3.1.9 Wband

SCPI Commands

SENSe:IQ:WBANd

class WbandCls

Wband commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:IQ:WBANd
value: bool = driver.applications.iqAnalyzer.sense.iq.wband.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:IQ:WBAND
driver.applications.iqAnalyzer.sense.iq.wband.set(state = False)
```

No command help available

param state

No help available

6.1.1.3.2 SwapIq

SCPI Commands

SENSe:SWAPiq

class SwapIqCls

SwapIq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWAPiq
value: bool = driver.applications.iqAnalyzer.sense.swapIq.get()
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

return

state: ON | 1 I and Q signals are interchanged Inverted sideband, Q+j*I OFF | 0 I and Q signals are not interchanged Normal sideband, I+j*Q

set(state: bool) → None

```
# SCPI: [SENSe]:SWAPiq
driver.applications.iqAnalyzer.sense.swapIq.set(state = False)
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

param state

ON | 1 I and Q signals are interchanged Inverted sideband, Q+j*I OFF | 0 I and Q signals are not interchanged Normal sideband, I+j*Q

6.1.1.4 Trace

class TraceCls

Trace commands group definition. 22 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.clone()
```

Subgroups

6.1.1.4.1 Iq

class IqCls

Iq commands group definition. 22 total commands, 13 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.clone()
```

Subgroups

6.1.1.4.1.1 Apcon

class ApconCls

Apcon commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.apcon.clone()
```

Subgroups

6.1.1.4.1.2 A

SCPI Commands

```
TRACe:IQ:APCon:A
```

class ACls

A commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:APCon:A
value: float = driver.applications.iqAnalyzer.trace.iq.apcon.a.get()
```

No command help available

```
return
    conversion_factor: No help available
```

set(*conversion_factor: float*) → None

```
# SCPI: TRACe:IQ:APCon:A
driver.applications.iqAnalyzer.trace.iq.apcon.a.set(conversion_factor = 1.0)
```

No command help available

```
param conversion_factor
    No help available
```

6.1.1.4.1.3 B

SCPI Commands

```
TRACe:IQ:APCon:B
```

class BCls

B commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:APCon:B
value: float = driver.applications.iqAnalyzer.trace.iq.apcon.b.get()
```

No command help available

```
return
    conversion_factor: No help available
```

set(*conversion_factor: float*) → None

```
# SCPI: TRACe:IQ:APCon:B
driver.applications.iqAnalyzer.trace.iq.apcon.b.set(conversion_factor = 1.0)
```

No command help available

```
param conversion_factor
    No help available
```

6.1.1.4.1.4 Result

SCPI Commands

TRACe:IQ:APCon:RESult

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:APCon:RESult
value: float = driver.applications.iqAnalyzer.trace.iq.apcon.result.get()
```

No command help available

```
return
    avg_power: No help available
```

6.1.1.4.1.5 State

SCPI Commands

TRACe:IQ:APCon:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:APCon[:STATe]
value: bool = driver.applications.iqAnalyzer.trace.iq.apcon.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: TRACe:IQ:APCon[:STATe]
driver.applications.iqAnalyzer.trace.iq.apcon.state.set(state = False)
```

No command help available

```
param state
    No help available
```

6.1.1.4.1.6 Average

class AverageCls

Average commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.average.clone()
```

Subgroups

6.1.1.4.1.7 Count

SCPI Commands

```
TRACe:IQ:AVERAge:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:AVERAge:COUNT
value: int = driver.applications.iqAnalyzer.trace.iq.average.count.get()
```

This command defines the number of I/Q data sets that the averaging is based on.

return

count: No help available

set(count: int) → None

```
# SCPI: TRACe:IQ:AVERAge:COUNT
driver.applications.iqAnalyzer.trace.iq.average.count.set(count = 1)
```

This command defines the number of I/Q data sets that the averaging is based on.

param count

Range: 0 to 32767

6.1.1.4.1.8 State

SCPI Commands

```
TRACe:IQ:AVERAge:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:AVERage[:STATe]
value: bool = driver.applications.iqAnalyzer.trace.iq.average.state.get()
```

This command turns averaging of the I/Q data on and off. Before you can use the command you have to turn the I/Q data acquisition on with method `Rsfswp.Applications.IqAnalyzer.Trace.Iq.State.set`. If averaging is on, the maximum amount of I/Q data that can be recorded is 512kS (524288 samples) .

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: TRACe:IQ:AVERage[:STATe]
driver.applications.iqAnalyzer.trace.iq.average.state.set(state = False)
```

This command turns averaging of the I/Q data on and off. Before you can use the command you have to turn the I/Q data acquisition on with method `Rsfswp.Applications.IqAnalyzer.Trace.Iq.State.set`. If averaging is on, the maximum amount of I/Q data that can be recorded is 512kS (524288 samples) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.1.4.1.9 Data

class DataCls

Data commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.data.clone()
```

Subgroups

6.1.1.4.1.10 FormatPy

SCPI Commands

```
TRACe:IQ:DATA:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → IqResultDataFormat

```
# SCPI: TRACe:IQ:DATA:FORMat
value: enums.IqResultDataFormat = driver.applications.iqAnalyzer.trace.iq.data.
    ↪ formatPy.get()
```


This command selects the order of the I/Q data. For details see ‘Reference: format description for I/Q data files’.

return

data_format: No help available

set(data_format: *IqResultDataFormat*) → None

```
# SCPI: TRACe:IQ:DATA:FORMat
driver.applications.iqAnalyzer.trace.iq.data.formatPy.set(data_format = enums.
↪ IqResultDataFormat.COMPAtible)
```

This command selects the order of the I/Q data. For details see ‘Reference: format description for I/Q data files’.

param data_format

COMPAtible | IQBLoCk | IQPair COMPAtible I and Q values are separated and collected in blocks: A block (512k) of I values is followed by a block (512k) of Q values, followed by a block of I values, followed by a block of Q values etc. (I,I,I,I,Q,Q,Q,Q,I,I,I,I,Q,Q,Q,Q...) IQBLoCk First all I-values are listed, then the Q-values (I,I,I,I,I,I... Q,Q,Q,Q,Q,Q) IQPair One pair of I/Q values after the other is listed (I,Q,I,Q,I,Q...).

6.1.1.4.1.11 DiqFilter

SCPI Commands

TRACe:IQ:DIQFilter

class DiqFilterCls

DiqFilter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:DIQFilter
value: bool = driver.applications.iqAnalyzer.trace.iq.diqFilter.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: TRACe:IQ:DIQFilter
driver.applications.iqAnalyzer.trace.iq.diqFilter.set(state = False)
```

No command help available

param state

No help available

6.1.1.4.1.12 Egate

class EgateCls

Egate commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.egate.clone()
```

Subgroups

6.1.1.4.1.13 Gap

SCPI Commands

```
TRACe:IQ:EGATe:GAP
```

class GapCls

Gap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:EGATe:GAP
value: int = driver.applications.iqAnalyzer.trace.iq.egate.gap.get()
```

This command defines the interval between several gate periods for gated measurements with the I/Q analyzer.

return

gap: No help available

set(gap: int) → None

```
# SCPI: TRACe:IQ:EGATe:GAP
driver.applications.iqAnalyzer.trace.iq.egate.gap.set(gap = 1)
```

This command defines the interval between several gate periods for gated measurements with the I/Q analyzer.

param gap

numeric value Max = (440 MS * sample rate/200MHz) -1 pretrigger samples defined by TRACe:IQ:SET; sample rate defined by method RsFswp.Applications.IqAnalyzer.Trace.Iq.SymbolRate.set) Range: 1...Max (samples)

6.1.1.4.1.14 Length

SCPI Commands

```
TRACe:IQ:EGATe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:EGATe:LENGth
value: int = driver.applications.iqAnalyzer.trace.iq.egate.length.get()
```

This command defines the gate length for gated measurements with the I/Q analyzer.

return
length: No help available

set(length: int) → None

```
# SCPI: TRACe:IQ:EGATe:LENGth
driver.applications.iqAnalyzer.trace.iq.egate.length.set(length = 1)
```

This command defines the gate length for gated measurements with the I/Q analyzer.

param length
numeric value Max = (440 MS * sample rate/200MHz) -1 pretrigger samples defined by TRACe:IQ:SET; sample rate defined by method RsFswp.Applications.IqAnalyzer.Trace.Iq.SymbolRate.set) Range: 1...Max (samples)

6.1.1.4.1.15 Nof

SCPI Commands

```
TRACe:IQ:EGATe:NOF
```

class NofCls

Nof commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:EGATe:NOF
value: int = driver.applications.iqAnalyzer.trace.iq.egate.nof.get()
```

This command defines the number of gate periods after the trigger signal for gated measurements with the I/Q analyzer.

return
periods_count: No help available

set(periods_count: int) → None

```
# SCPI: TRACe:IQ:EGATe:NOF
driver.applications.iqAnalyzer.trace.iq.egate.nof.set(periods_count = 1)
```

This command defines the number of gate periods after the trigger signal for gated measurements with the I/Q analyzer.

param periods_count

Range: 1 to 1023

6.1.1.4.1.16 State

SCPI Commands

TRACe:IQ:EGATe:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:EGATe[:STATe]
value: bool = driver.applications.iqAnalyzer.trace.iq.egate.state.get()
```

This command turns gated measurements with the I/Q analyzer on and off. Before you can use the command you have to turn on the I/Q analyzer and select an external or IF power trigger source.

return

state: ON | OFF

set(state: bool) → None

```
# SCPI: TRACe:IQ:EGATe[:STATe]
driver.applications.iqAnalyzer.trace.iq.egate.state.set(state = False)
```

This command turns gated measurements with the I/Q analyzer on and off. Before you can use the command you have to turn on the I/Q analyzer and select an external or IF power trigger source.

param state

ON | OFF

6.1.1.4.1.17 TypePy

SCPI Commands

TRACe:IQ:EGATe:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EgateType

```
# SCPI: TRACe:IQ:EGATe:TYPE
value: enums.EgateType = driver.applications.iqAnalyzer.trace.iq.egate.typePy.
    ↪get()
```

This command selects the gate mode for gated measurements with the I/Q analyzer. Note: The IF power trigger holdoff time is ignored if you are using the 'Level' gate mode in combination with an IF Power trigger.

```

    return
        type_py: LEVel EDGE

set(type_py: EgateType) → None

```

```

# SCPI: TRACe:IQ:EGATe:TYPE
driver.applications.iqAnalyzer.trace.iq.egate.typePy.set(type_py = enums.
↳EgateType.EDGE)

```

This command selects the gate mode for gated measurements with the I/Q analyzer. Note: The IF power trigger holdoff time is ignored if you are using the 'Level' gate mode in combination with an IF Power trigger.

```

    param type_py
        LEVel EDGE

```

6.1.1.4.1.18 Eval

SCPI Commands

```
TRACe:IQ:EVAL
```

class EvalCls

Eval commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get() → bool
```

```

# SCPI: TRACe:IQ:EVAL
value: bool = driver.applications.iqAnalyzer.trace.iq.eval.get()

```

This command turns I/Q data analysis on and off. Before you can use this command, you have to turn on the I/Q data acquisition using INST:CRE:NEW IQ or method RsFswp.Instrument.Create.Replace.set, or using the method RsFswp.Applications. IqAnalyzer.Trace.Iq.State.set command to replace the current channel while retaining the settings.

```

    return
        state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function
        on

set(state: bool) → None

```

```

# SCPI: TRACe:IQ:EVAL
driver.applications.iqAnalyzer.trace.iq.eval.set(state = False)

```

This command turns I/Q data analysis on and off. Before you can use this command, you have to turn on the I/Q data acquisition using INST:CRE:NEW IQ or method RsFswp.Instrument.Create.Replace.set, or using the method RsFswp.Applications. IqAnalyzer.Trace.Iq.State.set command to replace the current channel while retaining the settings.

```

    param state
        ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

```

6.1.1.4.1.19 File

class FileCls

File commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.file.clone()
```

Subgroups

6.1.1.4.1.20 Repetition

class RepetitionCls

Repetition commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.file.repetition.clone()
```

Subgroups

6.1.1.4.1.21 Count

SCPI Commands

```
TRACe:IQ:FILE:REPetition:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNt
value: int = driver.applications.iqAnalyzer.trace.iq.file.repetition.count.get()
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

return

repetitions: No help available

set(repetitions: int) → None

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNt
driver.applications.iqAnalyzer.trace.iq.file.repetition.count.set(repetitions = 1)
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

param repetitions
integer

6.1.1.4.1.22 Rlength

SCPI Commands

TRACe:IQ:RLENgth

class RlengthCls

Rlength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:RLENgth
value: int = driver.applications.iqAnalyzer.trace.iq.rlength.get()
```

This command sets the record length for the acquired I/Q data. Increasing the record length also increases the measurement time. Note: Alternatively, you can define the measurement time using the SENS:SWE:TIME command.

return
record_length: No help available

set(record_length: int) → None

```
# SCPI: TRACe:IQ:RLENgth
driver.applications.iqAnalyzer.trace.iq.rlength.set(record_length = 1)
```

This command sets the record length for the acquired I/Q data. Increasing the record length also increases the measurement time. Note: Alternatively, you can define the measurement time using the SENS:SWE:TIME command.

param record_length
Number of samples to record. See ‘Sample rate and maximum usable I/Q bandwidth for RF input’.

6.1.1.4.1.23 State

SCPI Commands

TRACe:IQ:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ[:STATe]
value: bool = driver.applications.iqAnalyzer.trace.iq.state.get()
```

This command activates the simple I/Q data acquisition mode (see ‘Activating I/Q analyzer measurements’)
. Executing this command also has the following effects:

INTRO_CMD_HELP: Prerequisites for this command

- The sweep, amplitude, input and trigger settings from the measurement are retained.
- All measurements are turned off.
- All traces are set to ‘Blank’ mode.
- The I/Q data analysis mode is turned off (TRAC:IQ:EVAL OFF) .

Note: To turn trace display back on or to enable the evaluation functions of the I/Q Analyzer, execute the TRAC:IQ:EVAL ON command (see method RsFswp.Applications.IqAnalyzer.Trace.Iq.Eval.set) .

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: TRACe:IQ[:STATe]
driver.applications.iqAnalyzer.trace.iq.state.set(state = False)
```

This command activates the simple I/Q data acquisition mode (see ‘Activating I/Q analyzer measurements’)
. Executing this command also has the following effects:

INTRO_CMD_HELP: Prerequisites for this command

- The sweep, amplitude, input and trigger settings from the measurement are retained.
- All measurements are turned off.
- All traces are set to ‘Blank’ mode.
- The I/Q data analysis mode is turned off (TRAC:IQ:EVAL OFF) .

Note: To turn trace display back on or to enable the evaluation functions of the I/Q Analyzer, execute the TRAC:IQ:EVAL ON command (see method RsFswp.Applications.IqAnalyzer.Trace.Iq.Eval.set) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.1.4.1.24 SymbolRate

SCPI Commands

TRACe:IQ:SRATe

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:SRATe
value: int = driver.applications.iqAnalyzer.trace.iq.symbolRate.get()
```

This command sets the final user sample rate for the acquired I/Q data. Thus, the user sample rate can be modified without affecting the actual data capturing settings on the R&S FSWP. Note: The smaller the user sample rate, the smaller the usable I/Q bandwidth, see ‘Sample rate and maximum usable I/Q bandwidth for RF input’.

return

count: No help available

set(count: int) → None

```
# SCPI: TRACe:IQ:SRATe
driver.applications.iqAnalyzer.trace.iq.symbolRate.set(count = 1)
```

This command sets the final user sample rate for the acquired I/Q data. Thus, the user sample rate can be modified without affecting the actual data capturing settings on the R&S FSWP. Note: The smaller the user sample rate, the smaller the usable I/Q bandwidth, see ‘Sample rate and maximum usable I/Q bandwidth for RF input’.

param count

The valid sample rates are described in ‘Sample rate and maximum usable I/Q bandwidth for RF input’. Unit: HZ

6.1.1.4.1.25 TpiSample

SCPI Commands

TRACe:IQ:TPISample

class TpiSampleCls

TpiSample commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:TPISample
value: float = driver.applications.iqAnalyzer.trace.iq.tpiSample.get()
```

This command queries the time offset between the sample start and the trigger event (trigger point in sample = TPIS). Since the R&S FSWP usually samples with a much higher sample rate than the specific application actually requires, the trigger point determined internally is much more precise than the one determined from the (downsampled) data in the application. Thus, the TPIS indicates the offset between the sample start

and the actual trigger event. This value can only be determined in triggered measurements using external or IFPower triggers, otherwise the value is 0.

return
tpis: numeric value Unit: s

6.1.1.4.1.26 Wband

class WbandCls

Wband commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.iqAnalyzer.trace.iq.wband.clone()
```

Subgroups

6.1.1.4.1.27 Mbwidth

SCPI Commands

```
TRACe:IQ:WBAND:MBWidth
```

class MbwidthCls

Mbwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: TRACe:IQ:WBAND:MBWidth
value: int = driver.applications.iqAnalyzer.trace.iq.wband.mbwidth.get()
```

Defines the maximum analysis bandwidth. Any value can be specified; the next higher fixed bandwidth is used.

return
max_bandwidth: No help available

set(max_bandwidth: int) → None

```
# SCPI: TRACe:IQ:WBAND:MBWidth
driver.applications.iqAnalyzer.trace.iq.wband.mbwidth.set(max_bandwidth = 1)
```

Defines the maximum analysis bandwidth. Any value can be specified; the next higher fixed bandwidth is used.

param max_bandwidth

80 MHz Restricts the analysis bandwidth to a maximum of 80 MHz. The bandwidth extension option R&S FSWP-B320 is deactivated. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to OFF. 160 MHz Restricts the analysis bandwidth to a maximum of 160 MHz. The

bandwidth extension option R&S FSWP-B320 is deactivated. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to ON. 160 MHz | MAX
The bandwidth extension option is activated. The currently available maximum bandwidth is allowed. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to ON. Unit: Hz

6.1.1.4.1.28 State

SCPI Commands

TRACe:IQ:WBAND:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:WBAND[:STATE]
value: bool = driver.applications.iqAnalyzer.trace.iq.wband.state.get()
```

This command determines whether the wideband provided by bandwidth extension options is used or not (if installed) .

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: TRACe:IQ:WBAND[:STATE]
driver.applications.iqAnalyzer.trace.iq.wband.state.set(state = False)
```

This command determines whether the wideband provided by bandwidth extension options is used or not (if installed) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.1.4.1.29 Wfilter

SCPI Commands

TRACe:IQ:WFIltEr

class WfilterCls

Wfilter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:WFIltEr
value: bool = driver.applications.iqAnalyzer.trace.iq.wfilter.get()
```

Activates a 200 MHz filter before the A/D converter, thus restricting the processed bandwidth to 200 MHz while using the wideband processing path in the R&S FSWP. For prerequisites see manual operation.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: TRACe:IQ:WFIltEr
driver.applications.iqAnalyzer.trace.iq.wfilter.set(state = False)
```

Activates a 200 MHz filter before the A/D converter, thus restricting the processed bandwidth to 200 MHz while using the wideband processing path in the R&S FSWP. For prerequisites see manual operation.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.2 K30_NoiseFigure

class K30_NoiseFigureCls

K30_NoiseFigure commands group definition. 256 total commands, 13 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30_NoiseFigure.clone()
```

Subgroups

6.1.2.1 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k30NoiseFigure.calculate.repcap_window_get()
driver.applications.k30NoiseFigure.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 66 total commands, 4 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.clone()
```

Subgroups

6.1.2.1.1 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k30NoiseFigure.calculate.deltaMarker.repcap_deltaMarker_get()
driver.applications.k30NoiseFigure.calculate.deltaMarker.repcap_deltaMarker_set(repcap.
↳DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 10 total commands, 8 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.deltaMarker.clone()
```

Subgroups

6.1.2.1.1.1 Aoff

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:AOFF
driver.applications.k30NoiseFigure.calculate.deltaMarker.aoff.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns off all delta markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.1.2 Maximum**class MaximumCls**

Maximum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.deltaMarker.maximum.clone()
```

Subgroups**6.1.2.1.1.3 Next****SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:NEXT
driver.applications.k30NoiseFigure.calculate.deltaMarker.maximum.next.
↪set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next positive peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.1.4 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum[:PEAK]
driver.applications.k30NoiseFigure.calculate.deltaMarker.maximum.peak.
↪set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.1.5 Minimum

class MinimumCls

Minimum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.deltaMarker.minimum.clone()
```

Subgroups

6.1.2.1.1.6 Next

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:NEXT
driver.applications.k30NoiseFigure.calculate.deltaMarker.minimum.next.
↪set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.1.7 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum[:PEAK]
driver.applications.k30NoiseFigure.calculate.deltaMarker.minimum.peak.
↪set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.1.8 Mreference

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:MREference

class MreferenceCls

Mreference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → int

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MREference
value: int = driver.applications.k30NoiseFigure.calculate.deltaMarker.
↳mreference.get(window = repcap.Window.Default, deltaMarker = repcap.
↳DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1. The reference may be another marker or the fixed reference.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

arg_0: No help available

set(arg_0: int, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MREference
driver.applications.k30NoiseFigure.calculate.deltaMarker.mreference.set(arg_0 =
↳1, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1. The reference may be another marker or the fixed reference.

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.2.1.1.9 State

SCPI Commands

CALCulate<Window>:DELTamarker<DeltaMarker>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
value: bool = driver.applications.k30NoiseFigure.calculate.deltaMarker.state.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
driver.applications.k30NoiseFigure.calculate.deltaMarker.state.set(state =
↳False, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.2.1.1.10 Trace

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:TRACe

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
value: float = driver.applications.k30NoiseFigure.calculate.deltaMarker.trace.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

trace: Trace number the marker is assigned to.

set(trace: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
driver.applications.k30NoiseFigure.calculate.deltaMarker.trace.set(trace = 1.0,
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

Trace number the marker is assigned to.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.2.1.1.11 X

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:X

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
value: float = driver.applications.k30NoiseFigure.calculate.deltaMarker.x.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

position: Numeric value that defines the marker position on the x-axis. Range: The value range and unit depend on the measurement and scale of the x-axis.

set(*position: float, window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
driver.applications.k30NoiseFigure.calculate.deltaMarker.x.set(position = 1.0,
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param position

Numeric value that defines the marker position on the x-axis. Range: The value range and unit depend on the measurement and scale of the x-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.2.1.12 Y

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace*: NoiseFigureResultCustom, *window*=Window.Default, *deltaMarker*=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:Y
value: float = driver.applications.k30NoiseFigure.calculate.deltaMarker.y.
↳ get(trace = enums.NoiseFigureResultCustom.CPCold, window = repcap.Window.
↳ Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param trace

CPCold Queries calibration power (cold) results. CPHot Queries calibration power (hot) results. CYFactor Queries calibration ‘y-factor’ results. GAIN Queries ‘gain’ results. NOISe Queries ‘noise figure’ results. PCOLd Queries power (cold) results. PHOT Queries power (hot) results. TEMPerature Queries ‘noise temperature’ results. YFACTOR Queries ‘y-factor’ results.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

result: Result at the position of the delta marker. The unit is variable and depends on the one you have currently set. Unit: DBM

6.1.2.1.2 Limit<LimitIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k30NoiseFigure.calculate.limit.repcap_limitIx_get()
driver.applications.k30NoiseFigure.calculate.limit.repcap_limitIx_set(repcap.LimitIx.Nr1)
```

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:DElete`

class LimitCls

Limit commands group definition. 18 total commands, 12 Subgroups, 1 group commands Repeated Capability: LimitIx, default value after init: LimitIx.Nr1

delete(*window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:DElete
driver.applications.k30NoiseFigure.calculate.limit.delete(window = repcap.
↳Window.Default, limitIx = repcap.LimitIx.Default)
```

This command deletes a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

delete_with_opc(*window=Window.Default, limitIx=LimitIx.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.limit.clone()
```

Subgroups

6.1.2.1.2.1 Active

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:ACTive`

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACTive
value: float = driver.applications.k30NoiseFigure.calculate.limit.active.
↳get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command queries the names of all active limit lines.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit_lines: String containing the names of all active limit lines in alphabetical order.

6.1.2.1.2.2 Clear**class ClearCls**

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.limit.clear.clone()
```

Subgroups**6.1.2.1.2.3 Immediate****SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:CLEar:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, limitIx=LimitIx.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CLEar[:IMMediate]
driver.applications.k30NoiseFigure.calculate.limit.clear.immediate.set(window =
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command deletes the result of the current limit check. The command works on all limit lines in all measurement windows at the same time.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.2.4 Comment

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:COMMENT`

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → str

```
# SCPI: CALCulate<n>:LIMit<li>:COMMENT
value: str = driver.applications.k30NoiseFigure.calculate.limit.comment.
↳get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines a comment for a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

comment: String containing the description of the limit line.

set(*comment: str, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:COMMENT
driver.applications.k30NoiseFigure.calculate.limit.comment.set(comment = '1',
↳window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines a comment for a limit line.

param comment

String containing the description of the limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.5 Control

class ControlCls

Control commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.limit.control.clone()
```

Subgroups

6.1.2.1.2.6 Data

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:CONTrol:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol[:DATA]
value: List[float] = driver.applications.k30NoiseFigure.calculate.limit.control.
↳data.get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the horizontal definition points of a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit_line_points: Variable number of x-axis values. Note that the number of horizontal values has to be the same as the number of vertical values set with method RsFswp.Calculate.Limit.Lower.Data.set or method RsFswp.Calculate.Limit.Upper.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: HZ

set(limit_line_points: List[float], window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol[:DATA]
driver.applications.k30NoiseFigure.calculate.limit.control.data.set(limit_line_
↳points = [1.1, 2.2, 3.3], window = repcap.Window.Default, limitIx = repcap.
↳LimitIx.Default)
```

This command defines the horizontal definition points of a limit line.

param limit_line_points

Variable number of x-axis values. Note that the number of horizontal values has to be the same as the number of vertical values set with method `RsFswp.Calculate.Limit.Lower.Data.set` or method `RsFswp.Calculate.Limit.Upper.Data.set`. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.7 Shift

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:CONTrol:SHIFt`

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*distance*: float, *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:SHIFt
driver.applications.k30NoiseFigure.calculate.limit.control.shift.set(distance = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete limit line horizontally. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Numeric value. The unit depends on the scale of the x-axis. Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.8 Copy

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:COPY`

class CopyCls

Copy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → int

```
# SCPI: CALCulate<n>:LIMit<li>:COPY
value: int = driver.applications.k30NoiseFigure.calculate.limit.copy.get(window=
↳ repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command copies a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

line: 1 to 8 number of the new limit line name String containing the name of the limit line.

set(line: int, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:COPY
driver.applications.k30NoiseFigure.calculate.limit.copy.set(line = 1, window =
↳ repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command copies a limit line.

param line

1 to 8 number of the new limit line name String containing the name of the limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.9 Fail

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:FAIL
```

class FailCls

Fail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:FAIL
value: bool = driver.applications.k30NoiseFigure.calculate.limit.fail.
↳ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command queries the result of a limit check in the specified window. To get a valid result, you have to perform a complete measurement with synchronization to the end of the measurement before reading out the result. This is only possible for single measurement mode.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

result: 0 PASS 1 FAIL

6.1.2.1.2.10 Lower

class LowerCls

Lower commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.limit.lower.clone()
```

Subgroups

6.1.2.1.2.11 Data

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer[:DATA]
value: List[float] = driver.applications.k30NoiseFigure.calculate.limit.lower.
    data.get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of a lower limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit_line_points: Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

set(*limit_line_points*: List[float], *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer[:DATA]
driver.applications.k30NoiseFigure.calculate.limit.lower.data.set(limit_line_
↪points = [1.1, 2.2, 3.3], window = repcap.Window.Default, limitIx = repcap.
↪LimitIx.Default)
```

This command defines the vertical definition points of a lower limit line.

param limit_line_points

Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.12 Shift

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:SHIFt
```

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*distance*: float, *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:SHIFt
driver.applications.k30NoiseFigure.calculate.limit.lower.shift.set(distance = 1.
↪0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete lower limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Defines the distance that the limit line moves. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.13 State

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:LOWer:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:STATe
value: bool = driver.applications.k30NoiseFigure.calculate.limit.lower.state.
↳get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns a lower limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:STATe
driver.applications.k30NoiseFigure.calculate.limit.lower.state.set(state =
↳False, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns a lower limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.14 Name

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:NAME

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → str

```
# SCPI: CALCulate<n>:LIMit<li>:NAME
value: str = driver.applications.k30NoiseFigure.calculate.limit.name.get(window=
↳ repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects a limit line that already exists or defines a name for a new limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

name: String containing the limit line name.

set(name: str, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:NAME
driver.applications.k30NoiseFigure.calculate.limit.name.set(name = '1', window=
↳ repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects a limit line that already exists or defines a name for a new limit line.

param name

String containing the limit line name.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.15 State

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:STATe
value: bool = driver.applications.k30NoiseFigure.calculate.limit.state.
↳get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns the limit check for a specific limit line on and off. To query the limit check result, use method **RsFswp.Calculate.Limit.Fail.get_**. Note that a new command exists to activate the limit check and define the trace to be checked in one step (see method **RsFswp.Calculate.Limit.Trace.Check.set**) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:STATe
driver.applications.k30NoiseFigure.calculate.limit.state.set(state = False,
↳window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns the limit check for a specific limit line on and off. To query the limit check result, use method **RsFswp.Calculate.Limit.Fail.get_**. Note that a new command exists to activate the limit check and define the trace to be checked in one step (see method **RsFswp.Calculate.Limit.Trace.Check.set**) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.16 Trace

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:TRACe

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe
value: List[float] = driver.applications.k30NoiseFigure.calculate.limit.trace.
    ↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command links a limit line to one or more traces. Note that this command is maintained for compatibility reasons only. Limit lines no longer need to be assigned to a trace explicitly. The trace to be checked can be defined directly (as a suffix) in the new command to activate the limit check (see method RsFswp.Calculate.Limit.Trace.Check.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

trace_limit: 1 to 4

set(trace_limit: List[float], window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe
driver.applications.k30NoiseFigure.calculate.limit.trace.set(trace_limit = [1.1,
    ↪ 2.2, 3.3], window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command links a limit line to one or more traces. Note that this command is maintained for compatibility reasons only. Limit lines no longer need to be assigned to a trace explicitly. The trace to be checked can be defined directly (as a suffix) in the new command to activate the limit check (see method RsFswp.Calculate.Limit.Trace.Check.set) .

param trace_limit

1 to 4

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.1.2.1.2.17 TypePy

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → NoiseFigureLimit

```
# SCPI: CALCulate<n>:LIMit<li>:TYPE
value: enums.NoiseFigureLimit = driver.applications.k30NoiseFigure.calculate.
↳limit.typePy.get(window = repcap.Window.Default, limitIx = repcap.LimitIx.
↳Default)
```

This command configures a limit line for a particular result type.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

result: NOISE | GAIN | TEMPerature | YFACTOR | ENR | PHOT | PCOLd GAIN Assigns the limit line to 'gain' results. NOISE Assigns the limit line to 'noise figure' results. PCOLd Assigns the limit line to power (cold) results. PHOT Assigns the limit line to power (hot) results. TEMPerature Assigns the limit line to 'noise temperature' results. YFACTOR Assigns the limit line to 'y-factor' results.

set(result: NoiseFigureLimit, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:TYPE
driver.applications.k30NoiseFigure.calculate.limit.typePy.set(result = enums.
↳NoiseFigureLimit.ENR, window = repcap.Window.Default, limitIx = repcap.
↳LimitIx.Default)
```

This command configures a limit line for a particular result type.

param result

NOISE | GAIN | TEMPerature | YFACTOR | ENR | PHOT | PCOLd GAIN Assigns the limit line to 'gain' results. NOISE Assigns the limit line to 'noise figure' results. PCOLd Assigns the limit line to power (cold) results. PHOT Assigns the limit line to power (hot) results. TEMPerature Assigns the limit line to 'noise temperature' results. YFACTOR Assigns the limit line to 'y-factor' results.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.18 Upper

class UpperCls

Upper commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.limit.upper.clone()
```

Subgroups

6.1.2.1.2.19 Data

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UPPer:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer[:DATA]
value: List[float] = driver.applications.k30NoiseFigure.calculate.limit.upper.
    ↪data.get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of an upper limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

limit_line_points: Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

set(limit_line_points: List[float], window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer[:DATA]
driver.applications.k30NoiseFigure.calculate.limit.upper.data.set(limit_line_
    ↪points = [1.1, 2.2, 3.3], window = repcap.Window.Default, limitIx = repcap.
    ↪LimitIx.Default)
```

This command defines the vertical definition points of an upper limit line.

param limit_line_points

Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.20 Shift

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:UPPer:SHIFt`

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*distance*: float, *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:SHIFt
driver.applications.k30NoiseFigure.calculate.limit.upper.shift.set(distance = 1.
↪0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete upper limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Defines the distance that the limit line moves.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.2.21 State

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:UPPer:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:STATe
value: bool = driver.applications.k30NoiseFigure.calculate.limit.upper.state.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns an upper limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:STATe
driver.applications.k30NoiseFigure.calculate.limit.upper.state.set(state = ↪
↪False, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command turns an upper limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.1.2.1.3 Marker<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k30NoiseFigure.calculate.marker.repcap_marker_get()
driver.applications.k30NoiseFigure.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 9 total commands, 7 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.marker.clone()
```

Subgroups

6.1.2.1.3.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:AOFF
driver.applications.k30NoiseFigure.calculate.marker.aoff.set(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command turns off all markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.3.2 Maximum

class MaximumCls

Maximum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.marker.maximum.clone()
```

Subgroups

6.1.2.1.3.3 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:NEXT
driver.applications.k30NoiseFigure.calculate.marker.maximum.next.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.3.4 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum[:PEAK]
driver.applications.k30NoiseFigure.calculate.marker.maximum.peak.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.3.5 Minimum**class MinimumCls**

Minimum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.marker.minimum.clone()
```

Subgroups**6.1.2.1.3.6 Next****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:NEXT
driver.applications.k30NoiseFigure.calculate.marker.minimum.next.set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.3.7 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum[:PEAK]
driver.applications.k30NoiseFigure.calculate.marker.minimum.peak.set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.1.3.8 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
value: bool = driver.applications.k30NoiseFigure.calculate.marker.state.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>[:STATE]
driver.applications.k30NoiseFigure.calculate.marker.state.set(state = False,
↪ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.1.2.1.3.9 Trace

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
value: float = driver.applications.k30NoiseFigure.calculate.marker.trace.
↪ get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than ‘Blank’. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

trace: 1 to 4 Trace number the marker is assigned to.

set(trace: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
driver.applications.k30NoiseFigure.calculate.marker.trace.set(trace = 1.0,
↪ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

1 to 4 Trace number the marker is assigned to.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.2.1.3.10 X

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X
value: float = driver.applications.k30NoiseFigure.calculate.marker.x.get(window,
↪ = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker. Note that markers have to be positioned on a discrete frequency that is part of the frequency list. If you set the marker on a frequency not included in the frequency list, the application positions the marker to the nearest frequency in the list (rounding up or down) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

position: Numeric value that defines the marker position on the x-axis. The unit depends on the result display. Range: The range depends on the current x-axis range. , Unit: Hz

set(position: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X
driver.applications.k30NoiseFigure.calculate.marker.x.set(position = 1.0,
↪ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker. Note that markers have to be positioned on a discrete frequency that is part of the frequency list. If you set the marker on a frequency not included in the frequency list, the application positions the marker to the nearest frequency in the list (rounding up or down) .

param position

Numeric value that defines the marker position on the x-axis. The unit depends on the result display. Range: The range depends on the current x-axis range. , Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.1.2.1.3.11 Y

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace: NoiseFigureResultCustom, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.applications.k30NoiseFigure.calculate.marker.y.get(trace,
↪ = enums.NoiseFigureResultCustom.CPCold, window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

result: Selects the result. CPCold Queries calibration power (cold) results. CPHot Queries calibration power (hot) results. CYFactor Queries calibration ‘y-factor’ results. GAIN Queries ‘gain’ results. NOISE Queries ‘noise figure’ results. NUNCertainty Queries the ‘noise figure’ uncertainty results. PCOLd Queries power (cold)

results. PHOT Queries power (hot) results. TEMPerature Queries ‘noise temperature’ results. YFACTOR Queries ‘y-factor’ results.

6.1.2.1.4 Uncertainty

class UncertaintyCls

Uncertainty commands group definition. 29 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.clone()
```

Subgroups

6.1.2.1.4.1 Common

SCPI Commands

```
CALCulate<Window>:UNCertainty:COMMON
```

class CommonCls

Common commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:UNCertainty:COMMON
value: bool = driver.applications.k30NoiseFigure.calculate.uncertainty.common.
↳get(window = repcap.Window.Default)
```

This command turns matching of the noise source characteristics used during calibration and measurement on and off. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMON OFF) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:COMMON
driver.applications.k30NoiseFigure.calculate.uncertainty.common.set(state =
↳False, window = repcap.Window.Default)
```

This command turns matching of the noise source characteristics used during calibration and measurement on and off. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMON OFF) .

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.2 Data

class DataCls

Data commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.data.clone()
```

Subgroups

6.1.2.1.4.3 Frequency

SCPI Commands

```
CALCulate<Window>:UNCertainty:DATA:FREquency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:DATA:FREquency
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.data.
↪frequency.get(window = repcap.Window.Default)
```

This command defines the frequency for which the uncertainty should be calculated. This command is available if you have turned automatic determination of the DUT characteristics off with method `RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Frequency.set`.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

frequency: Frequency of the DUT. Unit: HZ

set(frequency: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:DATA:FREquency
driver.applications.k30NoiseFigure.calculate.uncertainty.data.frequency.
↪set(frequency = 1.0, window = repcap.Window.Default)
```

This command defines the frequency for which the uncertainty should be calculated. This command is available if you have turned automatic determination of the DUT characteristics off with method `Rsfswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Frequency.set`.

param frequency

Frequency of the DUT. Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.4 Gain

SCPI Commands

CALCulate<Window>:UNCertainty:DATA:GAIN

class GainCls

Gain commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*) → float

<pre># SCPI: CALCulate<n>:UNCertainty:DATA:GAIN value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.data. ↪ gain.get(window = repcap.Window.Default)</pre>
--

This command defines the 'gain' of the DUT. This command is available if you have turned automatic determination of the DUT characteristics off with method `Rsfswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Gain.set`.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

gain: 'Gain' of the DUT. Unit: DB

set(gain: float, window=*Window.Default*) → None

<pre># SCPI: CALCulate<n>:UNCertainty:DATA:GAIN driver.applications.k30NoiseFigure.calculate.uncertainty.data.gain.set(gain = 1. ↪ 0, window = repcap.Window.Default)</pre>

This command defines the 'gain' of the DUT. This command is available if you have turned automatic determination of the DUT characteristics off with method `Rsfswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Gain.set`.

param gain

'Gain' of the DUT. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.5 Noise

SCPI Commands

`CALCulate<Window>:UNCertainty:DATA:NOISe`

class NoiseCls

Noise commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:DATA:NOISe
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.data.
↳noise.get(window = repcap.Window.Default)
```

This command defines the noise level of the DUT. This command is available if you have turned automatic determination of the DUT characteristics off with method `RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Results.set`.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

noise_level: Noise level of the DUT. Unit: DB

set(*noise_level: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:DATA:NOISe
driver.applications.k30NoiseFigure.calculate.uncertainty.data.noise.set(noise_
↳level = 1.0, window = repcap.Window.Default)
```

This command defines the noise level of the DUT. This command is available if you have turned automatic determination of the DUT characteristics off with method `RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Data.Results.set`.

param noise_level

Noise level of the DUT. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.6 Results

SCPI Commands

`CALCulate<Window>:UNCertainty:DATA:RESults`

class ResultsCls

Results commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:UNCertainty:DATA:RESults
value: bool = driver.applications.k30NoiseFigure.calculate.uncertainty.data.
↳ results.get(window = repcap.Window.Default)
```

This command turns automatic determination of the DUT characteristics for the calculation of the uncertainty on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | 1 The application calculates the uncertainty with the DUT characteristics (‘noise figure’, ‘gain’ and frequency) resulting from the ‘noise figure’ measurement. OFF | 0 The application calculates the uncertainty with the DUT characteristics (‘noise figure’, ‘gain’ and frequency) based on the values you have defined manually.

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:DATA:RESults
driver.applications.k30NoiseFigure.calculate.uncertainty.data.results.set(state_
↳ False, window = repcap.Window.Default)
```

This command turns automatic determination of the DUT characteristics for the calculation of the uncertainty on and off.

param state

ON | 1 The application calculates the uncertainty with the DUT characteristics (‘noise figure’, ‘gain’ and frequency) resulting from the ‘noise figure’ measurement. OFF | 0 The application calculates the uncertainty with the DUT characteristics (‘noise figure’, ‘gain’ and frequency) based on the values you have defined manually.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.7 Enr

class EnrCls

Enr commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.clone()
```

Subgroups

6.1.2.1.4.8 Calibration

class CalibrationCls

Calibration commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.calibration.clone()
```

Subgroups

6.1.2.1.4.9 Uncertainty

SCPI Commands

```
CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty
```

class UncertaintyCls

Uncertainty commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
↳ calibration.uncertainty.get(window = repcap.Window.Default)
```

This command defines the uncertainty of a calibration noise source. This command is available when [SENSe:]CORRection:ENR:COMMon and [SENSe:]CORRection:ENR:COMMon are off. If a smart noise source is used for calibration, the uncertainty values defined in the SNS table are used.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty. Unit: DB

set(uncertainty: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.calibration.
↳ uncertainty.set(uncertainty = 1.0, window = repcap.Window.Default)
```

This command defines the uncertainty of a calibration noise source. This command is available when [SENSe:]CORRection:ENR:COMMon and [SENSe:]CORRection:ENR:COMMon are off. If a smart noise source is used for calibration, the uncertainty values defined in the SNS table are used.

param uncertainty

Uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.calibration.
↳uncertainty.clone()
```

Subgroups**6.1.2.1.4.10 Cold****SCPI Commands**

```
CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty:COLD
```

class ColdCls

Cold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty:COLD
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
↳calibration.uncertainty.cold.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

uncertainty: No help available

set(uncertainty: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty:COLD
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.calibration.
↳uncertainty.cold.set(uncertainty = 1.0, window = repcap.Window.Default)
```

No command help available

param uncertainty

1..n

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.11 Hot

SCPI Commands

`CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty:HOT`

class HotCls

Hot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty:HOT
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
↳ calibration.uncertainty.hot.get(window = repcap.Window.Default)
```

This command defines the uncertainty of a calibration noise source. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Hot temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

set(*uncertainty: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:CALibration:UNCertainty:HOT
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.calibration.
↳ uncertainty.hot.set(uncertainty = 1.0, window = repcap.Window.Default)
```

This command defines the uncertainty of a calibration noise source. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off.

param uncertainty

Hot temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.12 Uncertainty

SCPI Commands

`CALCulate<Window>:UNCertainty:ENR:UNCertainty`

class UncertaintyCls

Uncertainty commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
←uncertainty.get(window = repcap.Window.Default)
```

This command defines the uncertainty of a noise source. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source. If a smart noise source is used, the uncertainty values defined in the SNS table are used.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty. Unit: DB

set(uncertainty: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.uncertainty.
←set(uncertainty = 1.0, window = repcap.Window.Default)
```

This command defines the uncertainty of a noise source. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source. If a smart noise source is used, the uncertainty values defined in the SNS table are used.

param uncertainty

Uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.uncertainty.clone()
```

Subgroups

6.1.2.1.4.13 Cold

SCPI Commands

```
CALCulate<Window>:UNCertainty:ENR:UNCertainty:COLD
```

class ColdCls

Cold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty:COLD
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
↳ uncertainty.cold.get(window = repcap.Window.Default)
```

This command defines the uncertainty of a resistor. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Cold temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

set(*uncertainty: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty:COLD
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.uncertainty.cold.
↳ set(uncertainty = 1.0, window = repcap.Window.Default)
```

This command defines the uncertainty of a resistor. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param uncertainty

Cold temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.14 Hot

SCPI Commands

```
CALCulate<Window>:UNCertainty:ENR:UNCertainty:HOT
```

class HotCls

Hot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty:HOT
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.enr.
↳ uncertainty.hot.get(window = repcap.Window.Default)
```

This command defines the uncertainty of a resistor. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Hot temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

set(*uncertainty: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:ENR:UNCertainty:HOT
driver.applications.k30NoiseFigure.calculate.uncertainty.enr.uncertainty.hot.
↪set(uncertainty = 1.0, window = repcap.Window.Default)
```

This command defines the uncertainty of a resistor. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param uncertainty

Hot temperature uncertainty value of the noise source. Refer to the data sheet of the noise source to determine its uncertainty.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.15 Match**class MatchCls**

Match commands group definition. 12 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.clone()
```

Subgroups**6.1.2.1.4.16 Dut****class DutCls**

Dut commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.clone()
```

Subgroups

6.1.2.1.4.17 InputPy

class InputPyCls

InputPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.inputPy.
↳ clone()
```

Subgroups

6.1.2.1.4.18 RI

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:DUT:IN:RL
```

class RLCls

RI commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:IN:RL
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳ dut.inputPy.rl.get(window = repcap.Window.Default)
```

This command defines the return loss at the DUT input.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

return_loss: Unit: DB

set(return_loss: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:IN:RL
driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.inputPy.rl.
↳ set(return_loss = 1.0, window = repcap.Window.Default)
```

This command defines the return loss at the DUT input.

param return_loss

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.19 Vswr

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:DUT:IN:VSWR
```

class VswrCls

Vswr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:IN[:VSWR]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
    dut.inputPy.vswr.get(window = repcap.Window.Default)
```

This command defines the VSWR at the DUT input.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

vswr: No help available

set(vswr: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:IN[:VSWR]
driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.inputPy.vswr.
    set(vswr = 1.0, window = repcap.Window.Default)
```

This command defines the VSWR at the DUT input.

param vswr

1..n

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.20 Out

class OutCls

Out commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.out.clone()
```

Subgroups

6.1.2.1.4.21 RI

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:DUT:OUT:RL
```

class RLCls

RI commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:OUT:RL
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳ dut.out.rl.get(window = repcap.Window.Default)
```

This command defines the returns loss at the DUT output.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

return_loss: Unit: DB

set(return_loss: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:OUT:RL
driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.out.rl.
↳ set(return_loss = 1.0, window = repcap.Window.Default)
```

This command defines the returns loss at the DUT output.

param return_loss

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.22 Vswr

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:DUT:OUT:VSWR
```

class VswrCls

Vswr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:OUT[:VSWR]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳ dut.out.vswr.get(window = repcap.Window.Default)
```

This command defines the VSWR at the DUT output.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

vswr: No help available

set(vswr: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:DUT:OUT[:VSWR]
driver.applications.k30NoiseFigure.calculate.uncertainty.match.dut.out.vswr.
↪set(vswr = 1.0, window = repcap.Window.Default)
```

This command defines the VSWR at the DUT output.

param vswr

1..n

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.23 Preamp

class PreampCls

Preamp commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.preamp.clone()
```

Subgroups

6.1.2.1.4.24 RI

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:PREamp:RL
```

class RLCls

RI commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:PREamp:RL
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↪preamp.rl.get(window = repcap.Window.Default)
```

This command defines the return loss at the input of the preamplifier.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

return_loss: Unit: DB

set(return_loss: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:PREamp:RL
driver.applications.k30NoiseFigure.calculate.uncertainty.match.preamp.rl.
↪set(return_loss = 1.0, window = repcap.Window.Default)
```

This command defines the return loss at the input of the preamplifier.

param return_loss

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.25 Vswr**SCPI Commands**

```
CALCulate<Window>:UNCertainty:MATCH:PREamp:VSWR
```

class VswrCls

Vswr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:PREamp[:VSWR]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↪preamp.vswr.get(window = repcap.Window.Default)
```

This command defines the VSWR at the input of the preamplifier. The command is available if you have turned on the preamplifier with method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Preamp.State.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

vswr: No help available

set(vswr: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:PREamp[:VSWR]
driver.applications.k30NoiseFigure.calculate.uncertainty.match.preamp.vswr.
↪set(vswr = 1.0, window = repcap.Window.Default)
```

This command defines the VSWR at the input of the preamplifier. The command is available if you have turned on the preamplifier with method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Preamp.State.set.

param vswr
1..n

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.26 Source

class SourceCls

Source commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.clone()
```

Subgroups

6.1.2.1.4.27 Calibration

class CalibrationCls

Calibration commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.
↳calibration.clone()
```

Subgroups

6.1.2.1.4.28 RI

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:RL
```

class RLCls

RI commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration:RL
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳source.calibration.rl.get(window = repcap.Window.Default)
```

This command defines the return loss at the calibration noise source output. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off. If a smart noise source is used, the return loss values defined in the SNS table are used.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

return_loss: Unit: DB

set(return_loss: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration:RL
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.
↪calibration.rl.set(return_loss = 1.0, window = repcap.Window.Default)
```

This command defines the return loss at the calibration noise source output. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off. If a smart noise source is used, the return loss values defined in the SNS table are used.

param return_loss

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.29 Sns

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:SNS
```

class SnsCls

Sns commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration:SNS
value: bool = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↪source.calibration.sns.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration:SNS
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.
↳ calibration.sns.set(state = False, window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.30 Vswr

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:VSWR
```

class VswrCls

Vswr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration[:VSWR]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳ source.calibration.vswr.get(window = repcap.Window.Default)
```

This command defines the VSWR at the calibration noise source output. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off. If a smart noise source is used, the VSWR values defined in the SNS table are used.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

vswr: No help available

set(vswr: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:CALibration[:VSWR]
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.
↳ calibration.vswr.set(vswr = 1.0, window = repcap.Window.Default)
```

This command defines the VSWR at the calibration noise source output. This command is available when [SENSe:]CORRection:ENR:COMMon and method RsFswp.Applications.K30_NoiseFigure.Calculate.Uncertainty.Common.set are off. If a smart noise source is used, the VSWR values defined in the SNS table are used.

param vswr

1..n

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.31 RI**SCPI Commands**

`CALCulate<Window>:UNCertainty:MATCH:SOURce:RL`

class RLCls

RI commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:RL
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳source.rl.get(window = repcap.Window.Default)
```

This command defines the return loss at the noise source output. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

return_loss: Unit: DB

set(*return_loss: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:RL
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.rl.
↳set(return_loss = 1.0, window = repcap.Window.Default)
```

This command defines the return loss at the noise source output. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source.

param return_loss

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.32 Sns**SCPI Commands**

`CALCulate<Window>:UNCertainty:MATCH:SOURce:SNS`

class SnsCls

Sns commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:SNS
value: bool = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳source.sns.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce:SNS
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.sns.
↳set(state = False, window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.33 Vswr

SCPI Commands

```
CALCulate<Window>:UNCertainty:MATCH:SOURce:VSWR
```

class VswrCls

Vswr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce[:VSWR]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.match.
↳source.vswr.get(window = repcap.Window.Default)
```

This command defines the VSWR at the noise source output. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source. If a smart noise source is used, the VSWR values defined in the SNS table are used.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

vswr: No help available

set(vswr: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:MATCH:SOURce[:VSWR]
driver.applications.k30NoiseFigure.calculate.uncertainty.match.source.vswr.
↪set(vswr = 1.0, window = repcap.Window.Default)
```

This command defines the VSWR at the noise source output. If the noise sources during calibration and measurement are different, the command defines the uncertainty of the measurement noise source. If a smart noise source is used, the VSWR values defined in the SNS table are used.

param vswr
1..n

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.34 Preamp

class PreampCls

Preamp commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.clone()
```

Subgroups

6.1.2.1.4.35 Gain

SCPI Commands

```
CALCulate<Window>:UNCertainty:PREamp:GAIN
```

class GainCls

Gain commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:GAIN
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.
↪gain.get(window = repcap.Window.Default)
```

This command define the ‘gain’ of an external preamplifier that may be part of the test setup.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

gain: Gain of the preamplifier. Refer to the data sheet of the preamplifier to determine its 'gain'. Unit: DB

set(gain: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:GAIN
driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.gain.set(gain = 1.0, window = repcap.Window.Default)
```

This command define the 'gain' of an external preamplifier that may be part of the test setup.

param gain

Gain of the preamplifier. Refer to the data sheet of the preamplifier to determine its 'gain'. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.2.1.4.36 Noise**SCPI Commands**

```
CALCulate<Window>:UNCertainty:PREamp:NOISe
```

class NoiseCls

Noise commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:NOISe
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.noise.get(window = repcap.Window.Default)
```

This command defines the noise level of an external preamplifier that may be part of the test setup.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

noise_level: Noise level of the preamplifier. Refer to the data sheet of the preamplifier to determine its noise level. Unit: DB

set(noise_level: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:NOISe
driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.noise.set(noise_level = 1.0, window = repcap.Window.Default)
```

This command defines the noise level of an external preamplifier that may be part of the test setup.

param noise_level

Noise level of the preamplifier. Refer to the data sheet of the preamplifier to determine its noise level. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.37 State**SCPI Commands**

`CALCulate<Window>:UNCertainty:PREamp:STATE`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:STATE
value: bool = driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.
↳ state.get(window = repcap.Window.Default)
```

This command includes or excludes an external preamplifier from the uncertainty calculation. If the test setup uses an external preamplifier, you also have to define its ‘noise figure’ and ‘gain’ values.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | OFF | 1 | 0

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:UNCertainty:PREamp:STATE
driver.applications.k30NoiseFigure.calculate.uncertainty.preamp.state.set(state_
↳ = False, window = repcap.Window.Default)
```

This command includes or excludes an external preamplifier from the uncertainty calculation. If the test setup uses an external preamplifier, you also have to define its ‘noise figure’ and ‘gain’ values.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.2.1.4.38 Result**SCPI Commands**

`CALCulate<Window>:UNCertainty:RESult`

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → float`

```
# SCPI: CALCulate<n>:UNCertainty[:RESult]
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.result.
←get(window = repcap.Window.Default)
```

This command queries the uncertainty of ‘noise figure’ results.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

uncertainty: Measurement uncertainty in dB.

6.1.2.1.4.39 Sanalyzer

class SanalyzerCls

Sanalyzer commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.sanalyzer.clone()
```

Subgroups

6.1.2.1.4.40 Gain

class GainCls

Gain commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.sanalyzer.gain.clone()
```

Subgroups

6.1.2.1.4.41 Uncertainty

SCPI Commands

```
CALCulate<Window>:UNCertainty:SANalyzer:GAIN:UNCertainty
```

class UncertaintyCls

Uncertainty commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → float`

```
# SCPI: CALCulate<n>:UNCertainty:SANalyzer:GAIN:UNCertainty
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.
↳sanalyzer.gain.uncertainty.get(window = repcap.Window.Default)
```

This command queries the uncertainty value of the spectrum analyzer's internal 'gain'.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

uncertainty: 'Gain' uncertainty of the spectrum analyzer in dB. Unit: DB

6.1.2.1.4.42 Noise

class NoiseCls

Noise commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.calculate.uncertainty.sanalyzer.noise.clone()
```

Subgroups

6.1.2.1.4.43 Uncertainty

SCPI Commands

```
CALCulate<Window>:UNCertainty:SANalyzer:NOISE:UNCertainty
```

class UncertaintyCls

Uncertainty commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → float`

```
# SCPI: CALCulate<n>:UNCertainty:SANalyzer:NOISE:UNCertainty
value: float = driver.applications.k30NoiseFigure.calculate.uncertainty.
↳sanalyzer.noise.uncertainty.get(window = repcap.Window.Default)
```

This command queries the uncertainty value of the spectrum analyzer's internal noise.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

uncertainty: 'Noise figure' uncertainty of the spectrum analyzer in dB. Unit: DB

6.1.2.2 Display

class DisplayCls

Display commands group definition. 17 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.clone()
```

Subgroups

6.1.2.2.1 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k30NoiseFigure.display.window.repcap_window_get()
driver.applications.k30NoiseFigure.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 17 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.clone()
```

Subgroups

6.1.2.2.1.1 Minfo

class MinfoCls

Minfo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.minfo.clone()
```

Subgroups

6.1.2.2.1.2 State

SCPI Commands

DISPlay:WINDow<Window>:MINFo:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
value: bool = driver.applications.k30NoiseFigure.display.window.minfo.state.
↳ get(window = repcap.Window.Default)
```

This command turns the marker information in all diagrams on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: ON | 1 Displays the marker information in the diagrams. OFF | 0 Hides the marker information in the diagrams.

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
driver.applications.k30NoiseFigure.display.window.minfo.state.set(state = False,
↳ window = repcap.Window.Default)
```

This command turns the marker information in all diagrams on and off.

param state

ON | 1 Displays the marker information in the diagrams. OFF | 0 Hides the marker information in the diagrams.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.2.2.1.3 Mtable

SCPI Commands

DISPlay:WINDow<Window>:MTABle

class MtableCls

Mtable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AutoMode

```
# SCPI: DISPLAY[:WINDOW<n>]:MTABLE
value: enums.AutoMode = driver.applications.k30NoiseFigure.display.window.
↳ mtable.get(window = repcap.Window.Default)
```

This command turns the marker table on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

display_mode: ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table.
AUTO Turns on the marker table if 3 or more markers are active.

set(display_mode: AutoMode, window=Window.Default) → None

```
# SCPI: DISPLAY[:WINDOW<n>]:MTABLE
driver.applications.k30NoiseFigure.display.window.mtable.set(display_mode =
↳ enums.AutoMode.AUTO, window = repcap.Window.Default)
```

This command turns the marker table on and off.

param display_mode

ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table. AUTO Turns on the marker table if 3 or more markers are active.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.2.2.1.4 Size

SCPI Commands

```
DISPlay:WINDow<Window>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → Size

```
# SCPI: DISPLAY[:WINDOW<n>]:SIZE
value: enums.Size = driver.applications.k30NoiseFigure.display.window.size.
↳ get(window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

size: LARGe Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

set(size: Size, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
driver.applications.k30NoiseFigure.display.window.size.set(size = enums.Size.
↳LARGe, window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param size

LARGe Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.2.2.1.5 Table

class TableCls

Table commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.table.clone()
```

Subgroups

6.1.2.2.1.6 Item

SCPI Commands

```
DISPlay:WINDow<Window>:TABLE:ITEM
```

class ItemCls

Item commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ItemStruct

Response structure. Fields:

- Items: enums.NoiseFigureResult: NOISe | GAIN | TEMPerature | YFACtor | ENR | PHOT | PCOLd | CYFactor | CPHot | CPCold | NUNCertainty For a list of possible parameter values (table items) see the parameter description of the [CMDLINK: TRACen[:DATA]? CMDLINK] command.
- State: bool: ON | OFF | 1 | 0

get(window=Window.Default) → ItemStruct

```
# SCPI: DISPlay[:WINDow<n>]:TABLE:ITEM
value: ItemStruct = driver.applications.k30NoiseFigure.display.window.table.
↪item.get(window = repcap.Window.Default)
```

This command selects the items displayed in the Result Table.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

structure: for return value, see the help for ItemStruct structure arguments.

set(items: NoiseFigureResult, state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TABLE:ITEM
driver.applications.k30NoiseFigure.display.window.table.item.set(items = enums.
↪NoiseFigureResult.CPCold, state = False, window = repcap.Window.Default)
```

This command selects the items displayed in the Result Table.

param items

NOISe | GAIN | TEMPerature | YFACtor | ENR | PHOT | PCOLd | CYFactor | CPHot | CPCold | NUNCertainty For a list of possible parameter values (table items) see the parameter description of the method **RsFswp.Trace.Data.get_** command.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.2.2.1.7 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.applications.k30NoiseFigure.display.window.trace.repcap_trace_get()
driver.applications.k30NoiseFigure.display.window.trace.repcap_trace_set(repcap.Trace.
↪Tr1)
```

class TraceCls

Trace commands group definition. 13 total commands, 8 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.clone()
```

Subgroups

6.1.2.2.1.8 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TraceModeH

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
value: enums.TraceModeH = driver.applications.k30NoiseFigure.display.window.
↳ trace.mode.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: No help available

set(mode: TraceModeH, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
driver.applications.k30NoiseFigure.display.window.trace.mode.set(mode = enums.
↳ TraceModeH.BLANK, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.9 Preset

SCPI Commands

DISPlay:WINDow<Window>:TRACe<Trace>:PRESet

class PresetCls

Preset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → SelectAll

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
value: enums.SelectAll = driver.applications.k30NoiseFigure.display.window.
↳ trace.preset.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

result_type: No help available

set(*result_type: SelectAll, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
driver.applications.k30NoiseFigure.display.window.trace.preset.set(result_type_
↳ = enums.SelectAll.ALL, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

No command help available

param result_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.10 Smoothing

class SmoothingCls

Smoothing commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.smoothing.clone()
```

Subgroups

6.1.2.2.1.11 Aperture

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SMOothing:APERture
```

class ApertureCls

Aperture commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing:APERture
value: float = driver.applications.k30NoiseFigure.display.window.trace.
↳smoothing.aperture.get(window = repcap.Window.Default, trace = repcap.Trace.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

aperture: No help available

set(aperture: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing:APERture
driver.applications.k30NoiseFigure.display.window.trace.smoothing.aperture.
↳set(aperture = 1.0, window = repcap.Window.Default, trace = repcap.Trace.
↳Default)
```

No command help available

param aperture

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.12 State**SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:SMOothing:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing[:STATe]
value: bool = driver.applications.k30NoiseFigure.display.window.trace.smoothing.
↳state.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(*state: bool, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing[:STATe]
driver.applications.k30NoiseFigure.display.window.trace.smoothing.state.
↳set(state = False, window = repcap.Window.Default, trace = repcap.Trace.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.13 State

SCPI Commands

`DISPlay:WINDow<Window>:TRACe<Trace>:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
value: bool = driver.applications.k30NoiseFigure.display.window.trace.state.
↳ get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(*state: bool, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
driver.applications.k30NoiseFigure.display.window.trace.state.set(state = False,
↳ window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.14 Symbols

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SYMBols
```

class SymbolsCls

Symbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SYMBols
value: bool = driver.applications.k30NoiseFigure.display.window.trace.symbols.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command turns symbols that represent the measurement points on a trace on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SYMBols
driver.applications.k30NoiseFigure.display.window.trace.symbols.set(state =
↳False, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command turns symbols that represent the measurement points on a trace on and off.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.15 Uncertainty

SCPI Commands

DISPlay:WINDow<Window>:TRACe<Trace>:UNCertainty

class UncertaintyCls

Uncertainty commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:UNCertainty
value: bool = driver.applications.k30NoiseFigure.display.window.trace.
↳uncertainty.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

If enabled, an additional trace is displayed indicating the measured trace values \pm the uncertainty values determined by the uncertainty calculator. This result is only useful for 'noise figure' measurements.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:UNCertainty
driver.applications.k30NoiseFigure.display.window.trace.uncertainty.set(state =
↳False, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

If enabled, an additional trace is displayed indicating the measured trace values \pm the uncertainty values determined by the uncertainty calculator. This result is only useful for 'noise figure' measurements.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.16 X

class XCls

X commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.x.clone()
```

Subgroups

6.1.2.2.1.17 Scale

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALe
```

class ScaleCls

Scale commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → FrequencyType

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]
value: enums.FrequencyType = driver.applications.k30NoiseFigure.display.window.
↳ trace.x.scale.get(window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

This command selects the type of frequency displayed on the x-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

frequency: RF|IF|LO IF Intermediary frequency, e.g. for measurements on frequency converting DUTs. RF Radio frequency.

set(frequency: FrequencyType, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]
driver.applications.k30NoiseFigure.display.window.trace.x.scale.set(frequency =
↳ enums.FrequencyType.IF, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

This command selects the type of frequency displayed on the x-axis.

param frequency

RF|IF|LO IF Intermediary frequency, e.g. for measurements on frequency converting DUTs. RF Radio frequency.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.18 Y**class YCls**

Y commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.y.clone()
```

Subgroups**6.1.2.2.1.19 Scale****class ScaleCls**

Scale commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.y.scale.clone()
```

Subgroups**6.1.2.2.1.20 Auto****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:AUTO
value: bool = driver.applications.k30NoiseFigure.display.window.trace.y.scale.
    ↪ auto.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command turns automatic scaling of the y-axis on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:AUTO
driver.applications.k30NoiseFigure.display.window.trace.y.scale.auto.set(state_
↪= False, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command turns automatic scaling of the y-axis on and off.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.21 Bottom

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:BOTTom
```

class BottomCls

Bottom commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:BOTTom
value: float = driver.applications.k30NoiseFigure.display.window.trace.y.scale.
↪bottom.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the bottom value of the y-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

level: The value ranges depend on the result display. Noise figure -75 dB to 75 dB

Noise temperature -999990000 K to 999990000 K Y-factor -200 dB to 200 dB Gain
 -75 dB to 75 dB Power (hot) -200 dBm to 200 dBm Power (cold) -200 dBm to 200
 dBm Unit: DB

set(level: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:BOTTom
driver.applications.k30NoiseFigure.display.window.trace.y.scale.bottom.
↪set(level = 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the bottom value of the y-axis.

param level

The value ranges depend on the result display. Noise figure -75 dB to 75 dB Noise
 temperature -999990000 K to 999990000 K Y-factor -200 dB to 200 dB Gain -75 dB
 to 75 dB Power (hot) -200 dBm to 200 dBm Power (cold) -200 dBm to 200 dBm Unit:
 DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface
 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface
 'Trace')

6.1.2.2.1.22 RefLevel

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel
value: float = driver.applications.k30NoiseFigure.display.window.trace.y.scale.
↪refLevel.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the reference level (for all traces in all windows) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface
 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface
 'Trace')

return

reference_level: Range: see datasheet , Unit: DBM

set(reference_level: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel
driver.applications.k30NoiseFigure.display.window.trace.y.scale.refLevel.
↪set(reference_level = 1.0, window = repcap.Window.Default, trace = repcap.
↪Trace.Default)
```

This command defines the reference level (for all traces in all windows) .

param reference_level

Range: see datasheet , Unit: DBM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.display.window.trace.y.scale.refLevel.clone()
```

Subgroups

6.1.2.2.1.23 Auto

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RLEVel:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:AUTO
value: bool = driver.applications.k30NoiseFigure.display.window.trace.y.scale.
↪refLevel.auto.get(window = repcap.Window.Default, trace = repcap.Trace.
↪Default)
```

This command turns automatic determination of the reference level on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:AUTO
driver.applications.k30NoiseFigure.display.window.trace.y.scale.refLevel.auto.
↪set(state = False, window = repcap.Window.Default, trace = repcap.Trace.
↪Default)
```

This command turns automatic determination of the reference level on and off.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.2.1.24 Top

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:TOP
```

class TopCls

Top commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:TOP
value: float = driver.applications.k30NoiseFigure.display.window.trace.y.scale.
↪top.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the top value of the y-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

level: The value ranges depend on the result display. Noise figure -75 dB to 75 dB
Noise temperature -999990000 K to 999990000 K Y-factor -200 dB to 200 dB Gain
-75 dB to 75 dB Power (hot) -200 dBm to 200 dBm Power (cold) -200 dBm to 200
dBm Unit: DB

set(level: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:TOP
driver.applications.k30NoiseFigure.display.window.trace.y.scale.top.set(level =
↪1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```


This command defines the top value of the y-axis.

param level

The value ranges depend on the result display. Noise figure -75 dB to 75 dB Noise temperature -999990000 K to 999990000 K Y-factor -200 dB to 200 dB Gain -75 dB to 75 dB Power (hot) -200 dBm to 200 dBm Power (cold) -200 dBm to 200 dBm Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.2.3 FormatPy

class FormatPyCls

FormatPy commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.formatPy.clone()
```

Subgroups

6.1.2.3.1 Dexport

class DexportCls

Dexport commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.formatPy.dexport.clone()
```

Subgroups

6.1.2.3.1.1 Cseparator

SCPI Commands

```
FORMat:DEXPort:CSEPARATOR
```

class CseparatorCls

Cseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FileSeparator

```
# SCPI: FORMat:DEXPort:CSEPARATOR
value: enums.FileSeparator = driver.applications.k30NoiseFigure.formatPy.
↳ dexport.cseparator.get()
```

No command help available

```
return
column_separator: No help available
```

set(column_separator: FileSeparator) → None

```
# SCPI: FORMat:DEXPort:CSEPARATOR
driver.applications.k30NoiseFigure.formatPy.dexport.cseparator.set(column_
↳ separator = enums.FileSeparator.COMMa)
```

No command help available

```
param column_separator
No help available
```

6.1.2.3.1.2 Dseparator

SCPI Commands

FORMat:DEXPort:DSEPARATOR

class DseparatorCls

Dseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Separator

```
# SCPI: FORMat:DEXPort:DSEPARATOR
value: enums.Separator = driver.applications.k30NoiseFigure.formatPy.dexport.
↳ dseparator.get()
```

This command selects the decimal separator for data exported in ASCII format.

```
return
separator: POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05.
POINT Uses a point as decimal separator, e.g. 4.05.
```

set(separator: Separator) → None

```
# SCPI: FORMat:DEXPort:DSEPARATOR
driver.applications.k30NoiseFigure.formatPy.dexport.dseparator.set(separator =
↳ enums.Separator.COMMa)
```

This command selects the decimal separator for data exported in ASCII format.

```
param separator
POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05. POINT Uses
a point as decimal separator, e.g. 4.05.
```

6.1.2.3.1.3 FormatPy

SCPI Commands

FORMat:DEXPort:FORMat

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FileFormat

```
# SCPI: FORMat:DEXPort:FORMat
value: enums.FileFormat = driver.applications.k30NoiseFigure.formatPy.dexport.
↪formatPy.get()
```

No command help available

```
return
    file_format: No help available
```

set(file_format: FileFormat) → None

```
# SCPI: FORMat:DEXPort:FORMat
driver.applications.k30NoiseFigure.formatPy.dexport.formatPy.set(file_format =
↪enums.FileFormat.CSV)
```

No command help available

```
param file_format
    No help available
```

6.1.2.3.1.4 Header

SCPI Commands

FORMat:DEXPort:HEADer

class HeaderCls

Header commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: FORMat:DEXPort:HEADer
value: bool = driver.applications.k30NoiseFigure.formatPy.dexport.header.get()
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

```
return
    state: ON | OFF | 0 | 1
```

set(state: bool) → None

```
# SCPI: FORMat:DEXPort:HEADer
driver.applications.k30NoiseFigure.formatPy.dexport.header.set(state = False)
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

param state
ON | OFF | 0 | 1

6.1.2.3.1.5 Traces

SCPI Commands

FORMat:DEXPort:TRACes

class TracesCls

Traces commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SelectionScope

```
# SCPI: FORMat:DEXPort:TRACes
value: enums.SelectionScope = driver.applications.k30NoiseFigure.formatPy.
↳ dexport.traces.get()
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set).

return

selection: SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

set(selection: SelectionScope) → None

```
# SCPI: FORMat:DEXPort:TRACes
driver.applications.k30NoiseFigure.formatPy.dexport.traces.set(selection =
↳ enums.SelectionScope.ALL)
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set).

param selection

SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

6.1.2.4 Initiate

class InitiateCls

Initiate commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.initiate.clone()
```

Subgroups

6.1.2.4.1 Continuous

SCPI Commands

```
INITiate:CONTinuous
```

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INITiate:CONTinuous
value: bool = driver.applications.k30NoiseFigure.initiate.continuous.get()
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

return

state: ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

set(state: bool) → None

```
# SCPI: INITiate:CONTinuous
driver.applications.k30NoiseFigure.initiate.continuous.set(state = False)
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

param state

ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

6.1.2.4.2 Immediate

SCPI Commands

INITiate:IMMediate

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k30NoiseFigure.initiate.immediate.set()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k30NoiseFigure.initiate.immediate.set_with_opc()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.5 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k30NoiseFigure.inputPy.repcap_inputIx_get()
driver.applications.k30NoiseFigure.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 10 total commands, 8 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.inputPy.clone()
```

Subgroups

6.1.2.5.1 Attenuation

SCPI Commands

```
INPut<InputIx>:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:ATTenuation
value: float = driver.applications.k30NoiseFigure.inputPy.attenuation.
↳get(inputIx = repcap.InputIx.Default)
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

attenuation: Range: see data sheet , Unit: DB

set(attenuation: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:ATTenuation
driver.applications.k30NoiseFigure.inputPy.attenuation.set(attenuation = 1.0,↳
↳inputIx = repcap.InputIx.Default)
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

param attenuation

Range: see data sheet , Unit: DB

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.2 Connector

SCPI Commands

```
INPut<InputIx>:CONNector
```

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → InputConnectorB

```
# SCPI: INPut<ip>:CONNector
value: enums.InputConnectorB = driver.applications.k30NoiseFigure.inputPy.
↳connector.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

input_connectors: No help available

set(*input_connectors: InputConnectorB, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:CONNector
driver.applications.k30NoiseFigure.inputPy.connector.set(input_connectors =
↳enums.InputConnectorB.AIQI, inputIx = repcap.InputIx.Default)
```

No command help available

param input_connectors

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.3 Coupling

SCPI Commands

```
INPut<InputIx>:COUpling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → CouplingTypeA

```
# SCPI: INPut<ip>:COUpling
value: enums.CouplingTypeA = driver.applications.k30NoiseFigure.inputPy.
↳coupling.get(inputIx = repcap.InputIx.Default)
```

This command selects the coupling type of the RF input.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

coupling_type: AC | DC AC AC coupling DC DC coupling

set(*coupling_type: CouplingTypeA, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:COUpling
driver.applications.k30NoiseFigure.inputPy.coupling.set(coupling_type = enums.
↳CouplingTypeA.AC, inputIx = repcap.InputIx.Default)
```

This command selects the coupling type of the RF input.

param coupling_type

AC | DC AC AC coupling DC DC coupling

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.4 Dpath

SCPI Commands

```
INPut<InputIx>:DPATH
```

class DpathCls

Dpath commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → AutoOrOff

```
# SCPI: INPut<ip>:DPATH
value: enums.AutoOrOff = driver.applications.k30NoiseFigure.inputPy.dpath.
↳get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(*state: AutoOrOff, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:DPATH
driver.applications.k30NoiseFigure.inputPy.dpath.set(state = enums.AutoOrOff.
↳AUTO, inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.5 FilterPy**class FilterPyCls**

FilterPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.inputPy.filterPy.clone()
```

Subgroups**6.1.2.5.5.1 Hpass****class HpassCls**

Hpass commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.inputPy.filterPy.hpass.clone()
```

Subgroups**6.1.2.5.5.2 State****SCPI Commands**

```
INPut<InputIx>:FILTer:HPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:FILTer:HPASs[:STATe]
value: bool = driver.applications.k30NoiseFigure.inputPy.filterPy.hpass.state.
↳ get(inputIx = repcap.InputIx.Default)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:FILTer:HPASs[:STATe]
driver.applications.k30NoiseFigure.inputPy.filterPy.hpass.state.set(state = False, inputIx = repcap.InputIx.Default)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.5.3 Yig**class YigCls**

Yig commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.inputPy.filterPy.yig.clone()
```

Subgroups**6.1.2.5.5.4 State****SCPI Commands**

```
INPut<InputIx>:FILTer:YIG:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:FILTer:YIG[:STATe]
value: bool = driver.applications.k30NoiseFigure.inputPy.filterPy.yig.state.
↪ get(inputIx = repcap.InputIx.Default)
```

Enables or disables the YIG filter.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: ON | OFF | 0 | 1

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:FILTer:YIG[:STATe]
driver.applications.k30NoiseFigure.inputPy.filterPy.yig.state.set(state = False,
↪ inputIx = repcap.InputIx.Default)
```

Enables or disables the YIG filter.

param state

ON | OFF | 0 | 1

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.2.5.6 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.inputPy.gain.clone()
```

Subgroups

6.1.2.5.6.1 State

SCPI Commands

```
INPut<InputIx>:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → bool

```
# SCPI: INPut<ip>:GAIN:STATe
value: bool = driver.applications.k30NoiseFigure.inputPy.gain.state.get(inputIx,
↳= repcap.InputIx.Default)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:GAIN:STATe
driver.applications.k30NoiseFigure.inputPy.gain.state.set(state = False,
↳inputIx = repcap.InputIx.Default)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.2.5.6.2 Value

SCPI Commands

```
INPut<InputIx>:GAIN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → float

```
# SCPI: INPut<ip>:GAIN[:VALue]
value: float = driver.applications.k30NoiseFigure.inputPy.gain.value.
↳get(inputIx = repcap.InputIx.Default)
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

gain: For R&S FSWP8 and R&S FSWP26, the following settings are available: 15 dB and 30 dB All other values are rounded to the nearest of these two. R&S FSWP50: 30 dB Unit: DB

set(gain: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:GAIN[:VALue]
driver.applications.k30NoiseFigure.inputPy.gain.value.set(gain = 1.0, inputIx =
↳repcap.InputIx.Default)
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications. K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param gain

For R&S FSWP8 and R&S FSWP26, the following settings are available: 15 dB and 30 dB All other values are rounded to the nearest of these two. R&S FSWP50: 30 dB Unit: DB

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.2.5.7 Impedance

SCPI Commands

```
INPut<InputIx>:IMPedance
```

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → int

```
# SCPI: INPut<ip>:IMPedance
value: int = driver.applications.k30NoiseFigure.inputPy.impedance.get(inputIx =
↳repcap.InputIx.Default)
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

impedance: 50 | 75 Unit: OHM

set(impedance: int, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IMPedance
driver.applications.k30NoiseFigure.inputPy.impedance.set(impedance = 1, inputIx=
↳ repcap.InputIx.Default)
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

param impedance
50 | 75 Unit: OHM

param inputIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.5.8 TypePy

SCPI Commands

```
INPut<InputIx>:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → InputSelect

```
# SCPI: INPut<ip>:TYPE
value: enums.InputSelect = driver.applications.k30NoiseFigure.inputPy.typePy.
↳ get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return
input_py: No help available

set(input_py: InputSelect, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:TYPE
driver.applications.k30NoiseFigure.inputPy.typePy.set(input_py = enums.
↳ InputSelect.INPut1, inputIx = repcap.InputIx.Default)
```

No command help available

param input_py
No help available

param inputIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.2.6 Layout

class LayoutCls

Layout commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.clone()
```

Subgroups

6.1.2.6.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.add.clone()
```

Subgroups

6.1.2.6.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK30) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k30NoiseFigure.layout.add.window.get(window_
↳ name = '1', direction = enums.WindowDirection.ABOVE, window_type = enums.
↳ WindowTypeK30.CalPowerCold=CPCold)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method `RsFswp.Layout.Replace.Window.set` command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_query**.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.2.6.2 Catalog**class CatalogCls**

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.catalog.clone()
```

Subgroups**6.1.2.6.2.1 Window****SCPI Commands**

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k30NoiseFigure.layout.catalog.window.  
↪get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return

result: No help available

6.1.2.6.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.identify.clone()
```

Subgroups

6.1.2.6.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k30NoiseFigure.layout.identify.window.
    ↪get(window_name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name

String containing the name of a window.

return

window_index: Index number of the window.

6.1.2.6.4 Move

class MoveCls

Move commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.move.clone()
```

Subgroups

6.1.2.6.4.1 Window

SCPI Commands

```
LAYout:MOVE:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(source_window: str, target_window: str, direction: WindowDirReplace) → None

```
# SCPI: LAYout:MOVE[:WINDow]
driver.applications.k30NoiseFigure.layout.move.window.set(source_window = '1',
↳target_window = '1', direction = enums.WindowDirReplace.ABOVE)
```

No command help available

param source_window

No help available

param target_window

No help available

param direction

No help available

6.1.2.6.5 Remove

class RemoveCls

Remove commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.remove.clone()
```

Subgroups

6.1.2.6.5.1 Window

SCPI Commands

LAYout:REMove:WINDow

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str) → None

```
# SCPI: LAYout:REMove[:WINDow]
driver.applications.k30NoiseFigure.layout.remove.window.set(window_name = '1')
```

This command removes a window from the display in the active channel.

param window_name

String containing the name of the window. In the default state, the name of the window is its index.

6.1.2.6.6 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.layout.replace.clone()
```

Subgroups

6.1.2.6.6.1 Window

SCPI Commands

LAYout:REPLace:WINDow

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeK30) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k30NoiseFigure.layout.replace.window.set(window_name = '1',
↪window_type = enums.WindowTypeK30.CalPowerCold=CPCold)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method `RsFswp.Layout.Add.Window.get_` command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method `RsFswp.Layout.Catalog.Window.get_` query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method `RsFswp.Layout.Add.Window.get_` for a list of available window types.

6.1.2.6.7 Splitter

SCPI Commands

LAYout:SPLitter

class SplitterCls

Splitter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SplitterStruct

Response structure. Fields:

- Index_1: float: The index of one window the splitter controls.
- Index_2: float: The index of a window on the other side of the splitter.
- Position: float: New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu) . The point of origin (x = 0, y = 0) is in the lower left corner of the screen. The end point (x = 100, y = 100) is in the upper right corner of the screen. (See Figure ‘SmartGrid coordinates for remote control of the splitters’.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

`get()` → SplitterStruct

<pre># SCPI: LAYout:SPLitter value: SplitterStruct = driver.applications.k30NoiseFigure.layout.splitter.get()</pre>

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method `RsFswp.Applications.K30_NoiseFigure.Display.Window.Size.set` command, the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

return

structure: for return value, see the help for SplitterStruct structure arguments.

`set(index_1: float, index_2: float, position: float) → None`

```
# SCPI: LAYout:SPLitter
driver.applications.k30NoiseFigure.layout.splitter.set(index_1 = 1.0, index_2 = 1.0, position = 1.0)
```

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method `Rsfswp.Applications.K30_NoiseFigure.Display.Window.Size.set` command, the method `Rsfswp.Applications.K30_NoiseFigure.Layout.Splitter.set` changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

param index_1

The index of one window the splitter controls.

param index_2

The index of a window on the other side of the splitter.

param position

New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu). The point of origin ($x = 0$, $y = 0$) is in the lower left corner of the screen. The end point ($x = 100$, $y = 100$) is in the upper right corner of the screen. (See Figure ‘SmartGrid coordinates for remote control of the splitters’.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

6.1.2.7 MassMemory

class MassMemoryCls

MassMemory commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.massMemory.clone()
```

Subgroups

6.1.2.7.1 Store<Store>

RepCap Settings

```
# Range: Pos1 .. Pos32
rc = driver.applications.k30NoiseFigure.massMemory.store.repcap_store_get()
driver.applications.k30NoiseFigure.massMemory.store.repcap_store_set(repcap.Store.Pos1)
```

class StoreCls

Store commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Store, default value after init: Store.Pos1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.massMemory.store.clone()
```

Subgroups

6.1.2.7.1.1 Trace

SCPI Commands

```
MMEMory:STORe<Store>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(trace: float, filename: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:TRACe
driver.applications.k30NoiseFigure.massMemory.store.trace.set(trace = 1.0,
↪filename = '1', store = repcap.Store.Default)
```

This command exports trace data from the specified window to an ASCII file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a ‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device.

param trace

Number of the trace to be stored

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.1.2.8 Output<OutputConnector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.applications.k30NoiseFigure.output.repcap_outputConnector_get()
driver.applications.k30NoiseFigure.output.repcap_outputConnector_set(repcap.
↪OutputConnector.Nr1)
```

class OutputCls

Output commands group definition. 5 total commands, 1 Subgroups, 0 group commands Repeated Capability: OutputConnector, default value after init: OutputConnector.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.output.clone()
```

Subgroups

6.1.2.8.1 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k30NoiseFigure.output.trigger.repcap_triggerPort_get()
driver.applications.k30NoiseFigure.output.trigger.repcap_triggerPort_set(repcap.
↳TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 4 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.output.trigger.clone()
```

Subgroups

6.1.2.8.1.1 Direction

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → InOutDirection

```
# SCPI: OUTPut<up>:TRIGger<tp>:DIRection
value: enums.InOutDirection = driver.applications.k30NoiseFigure.output.trigger.
↳direction.get(outputConnector = repcap.OutputConnector.Default, triggerPort =
↳repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

direction: INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

set(*direction: InOutDirection, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:DIRection
driver.applications.k30NoiseFigure.output.trigger.direction.set(direction =
↳ enums.InOutDirection.INPUT, outputConnector = repcap.OutputConnector.Default,
↳ triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param direction

INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.2.8.1.2 Level

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → LowHigh

```
# SCPI: OUTPut<up>:TRIGger<tp>:LEVel
value: enums.LowHigh = driver.applications.k30NoiseFigure.output.trigger.level.
↳ get(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↳ TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return
level: HIGH 5 V LOW 0 V

set(level: LowHigh, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:LEVel
driver.applications.k30NoiseFigure.output.trigger.level.set(level = enums.
↳LowHigh.HIGH, outputConnector = repcap.OutputConnector.Default, triggerPort =
↳repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param level
HIGH 5 V LOW 0 V

param outputConnector
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.2.8.1.3 Otype

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:OTYPE
```

class OtypeCls

Otype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → TriggerOutType

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPE
value: enums.TriggerOutType = driver.applications.k30NoiseFigure.output.trigger.
↳otype.get(outputConnector = repcap.OutputConnector.Default, triggerPort =
↳repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param outputConnector
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return
output_type: DEVICE Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEFined Sends a user-defined trigger signal. For more information, see method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set.

```
set(output_type: TriggerOutType, outputConnector=OutputConnector.Default,
    triggerPort=TriggerPort.Default) → None
```

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPe
driver.applications.k30NoiseFigure.output.trigger.otype.set(output_type = enums.
    ↳TriggerOutType.DEVICE, outputConnector = repcap.OutputConnector.Default,
    ↳triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param output_type

DEVICE Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEfined Sends a user-defined trigger signal. For more information, see method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.2.8.1.4 Pulse

class PulseCls

Pulse commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.output.trigger.pulse.clone()
```

Subgroups

6.1.2.8.1.5 Immediate

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
set(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None
```

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:IMMediate
driver.applications.k30NoiseFigure.output.trigger.pulse.immediate.
    ↳set(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
    ↳TriggerPort.Default)
```

This command generates a pulse at the trigger output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

set_with_opc(*outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default, opc_timeout_ms: int = -1*) → None

6.1.2.8.1.6 Length

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → float

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:LENGth
value: float = driver.applications.k30NoiseFigure.output.trigger.pulse.length.
↪ get(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↪ TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

length: Pulse length in seconds. Unit: S

set(*length: float, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:LENGth
driver.applications.k30NoiseFigure.output.trigger.pulse.length.set(length = 1.0,
↪ outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↪ TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param length

Pulse length in seconds. Unit: S

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.2.9 Sense**class SenseCls**

Sense commands group definition. 87 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.clone()
```

Subgroups**6.1.2.9.1 Bandwidth****class BandwidthCls**

Bandwidth commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.bandwidth.clone()
```

Subgroups**6.1.2.9.1.1 ListPy****class ListPyCls**

ListPy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.bandwidth.listPy.clone()
```

Subgroups

6.1.2.9.1.2 Data

SCPI Commands

SENSe:BWIDth:LIST:DATA

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: No parameter help available
- Bandwidth: List[float]: No parameter help available
- Sweep_Time: List[float]: No parameter help available

get() → DataStruct

```
# SCPI: [SENSe]:BWIDth:LIST:DATA
value: DataStruct = driver.applications.k30NoiseFigure.sense.bandwidth.listPy.
↳data.get()
```

No command help available

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], bandwidth: List[float], sweep_time: List[float]) → None

```
# SCPI: [SENSe]:BWIDth:LIST:DATA
driver.applications.k30NoiseFigure.sense.bandwidth.listPy.data.set(frequency =
↳[1.1, 2.2, 3.3], bandwidth = [1.1, 2.2, 3.3], sweep_time = [1.1, 2.2, 3.3])
```

No command help available

param frequency

No help available

param bandwidth

No help available

param sweep_time

No help available

6.1.2.9.2 Configure

class ConfigureCls

Configure commands group definition. 12 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.clone()
```

Subgroups

6.1.2.9.2.1 Control

SCPI Commands

```
SENSe:CONFigure:CONTRol
```

class ControlCls

Control commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:CONFigure:CONTRol
value: enums.AutoManualMode = driver.applications.k30NoiseFigure.sense.
    ↪ configure.control.get()
```

This command selects the measurement mode for the hot and cold power measurements. Note that selecting a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE or [SENSe:]CORRection:ENR[:MEASurement]:TYPE automatically selects manual measurement mode.

return

mode: AUTO | MANual AUTO Performs the Power (Hot) and Power (Cold) measurement in one step. MANual Performs the Power (Hot) and Power (Cold) measurement in two separate steps.

set(mode: AutoManualMode) → None

```
# SCPI: [SENSe]:CONFigure:CONTRol
driver.applications.k30NoiseFigure.sense.configure.control.set(mode = enums.
    ↪ AutoManualMode.AUTO)
```

This command selects the measurement mode for the hot and cold power measurements. Note that selecting a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE or [SENSe:]CORRection:ENR[:MEASurement]:TYPE automatically selects manual measurement mode.

param mode

AUTO | MANual AUTO Performs the Power (Hot) and Power (Cold) measurement in one step. MANual Performs the Power (Hot) and Power (Cold) measurement in two separate steps.

6.1.2.9.2.2 Correction

SCPI Commands

`SENSe:CONFigure:CORRection`

class CorrectionCls

Correction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CONFigure:CORRection
driver.applications.k30NoiseFigure.sense.configure.correction.set()
```

This command configures the software to perform calibration measurements. Using method RsFswp.Applications.K30_NoiseFigure.Initiate.Immediate.set then initiates a calibration instead of the actual measurement, until you deliberately select one of the normal measurements again with one of the following commands.

INTRO_CMD_HELP: Prerequisites for this command

- [SENSe:]CONFigure:FREQuency:CONTInuous
- [SENSe:]CONFigure:FREQuency:SINGle
- [SENSe:]CONFigure:LIST:CONTInuous
- [SENSe:]CONFigure:LIST:SINGle

Note that calibration data is used only when the second stage correction mode has been turned on with [SENSe:]CORRection[:STATe].

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CONFigure:CORRection
driver.applications.k30NoiseFigure.sense.configure.correction.set_with_opc()
```

This command configures the software to perform calibration measurements. Using method RsFswp.Applications.K30_NoiseFigure.Initiate.Immediate.set then initiates a calibration instead of the actual measurement, until you deliberately select one of the normal measurements again with one of the following commands.

INTRO_CMD_HELP: Prerequisites for this command

- [SENSe:]CONFigure:FREQuency:CONTInuous
- [SENSe:]CONFigure:FREQuency:SINGle
- [SENSe:]CONFigure:LIST:CONTInuous
- [SENSe:]CONFigure:LIST:SINGle

Note that calibration data is used only when the second stage correction mode has been turned on with [SENSe:]CORRection[:STATe].

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.2.3 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.frequency.clone()
```

Subgroups

6.1.2.9.2.4 Continuous

SCPI Commands

```
SENSe:CONFigure:FREQuency:CONTInuous
```

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CONFigure:FREQuency:CONTInuous
driver.applications.k30NoiseFigure.sense.configure.frequency.continuous.set()
```

This command configures the software to perform a single frequency measurement in continuous sweep mode.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CONFigure:FREQuency:CONTInuous
driver.applications.k30NoiseFigure.sense.configure.frequency.continuous.set_
↳ with_opc()
```

This command configures the software to perform a single frequency measurement in continuous sweep mode.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.2.5 Single

SCPI Commands

`SENSe:CONFigure:FREQuency:SINgle`

class SingleCls

Single commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`set()` → None

```
# SCPI: [SENSe]:CONFigure:FREQuency:SINgle
driver.applications.k30NoiseFigure.sense.configure.frequency.single.set()
```

This command configures the software to perform a single frequency measurement in single sweep mode.

`set_with_opc(opc_timeout_ms: int = -1)` → None

```
# SCPI: [SENSe]:CONFigure:FREQuency:SINgle
driver.applications.k30NoiseFigure.sense.configure.frequency.single.set_with_
↳opc()
```

This command configures the software to perform a single frequency measurement in single sweep mode.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.2.6 ListPy

class ListPyCls

ListPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.listPy.clone()
```

Subgroups

6.1.2.9.2.7 Continuous

SCPI Commands

`SENSe:CONFigure:LIST:CONTInuous`

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CONFigure:LIST:CONTinuous
driver.applications.k30NoiseFigure.sense.configure.listPy.continuous.set()
```

This command configures the software to perform a frequency list measurement in continuous sweep mode.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CONFigure:LIST:CONTinuous
driver.applications.k30NoiseFigure.sense.configure.listPy.continuous.set_with_
    ↪opc()
```

This command configures the software to perform a frequency list measurement in continuous sweep mode.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.2.8 Single

SCPI Commands

```
SENSe:CONFigure:LIST:SINgle
```

class SingleCls

Single commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CONFigure:LIST:SINgle
driver.applications.k30NoiseFigure.sense.configure.listPy.single.set()
```

This command configures the software to perform a measurement in single frequency tuning mode.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CONFigure:LIST:SINgle
driver.applications.k30NoiseFigure.sense.configure.listPy.single.set_with_opc()
```

This command configures the software to perform a measurement in single frequency tuning mode.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.2.9 Measurement

SCPI Commands

`SENSe:CONFigure:MEASurement`

class MeasurementCls

Measurement commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Temperature

```
# SCPI: [SENSe]:CONFigure:MEASurement
value: enums.Temperature = driver.applications.k30NoiseFigure.sense.configure.
↳ measurement.get()
```

This command selects the type of power measurement to perform next. The command is available for manual measurements (see[SENSe:]CONFigure:CONTRol).

return

measurement: HOT | COLD COLD Performs the Power (Cold) measurement next.
HOT Performs the Power (Hot) measurement next.

set(measurement: Temperature) → None

```
# SCPI: [SENSe]:CONFigure:MEASurement
driver.applications.k30NoiseFigure.sense.configure.measurement.set(measurement.
↳ = enums.Temperature.COLD)
```

This command selects the type of power measurement to perform next. The command is available for manual measurements (see[SENSe:]CONFigure:CONTRol).

param measurement

HOT | COLD COLD Performs the Power (Cold) measurement next. HOT Performs the Power (Hot) measurement next.

6.1.2.9.2.10 Mode

class ModeCls

Mode commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.mode.clone()
```

Subgroups

6.1.2.9.2.11 Dut

SCPI Commands

```
SENSe:CONFigure:MODE:DUT
```

class DutCls

Dut commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DutType

```
# SCPI: [SENSe]:CONFigure:MODE:DUT
value: enums.DutType = driver.applications.k30NoiseFigure.sense.configure.mode.
    dut.get()
```

This command selects the type of DUT you are testing. Note that you have to use [SENSe:]CONFigure:MODE:SYSTem:LO to select if the LO or IF are fixed.

return

dut_type: AMPLifier | DDOWnconv | DOWNnconv | UPConv AMPLifier Measurements on fixed frequency DUTs. DOWNnconv Measurements on down-converting DUTs. UPConv Measurements on up-converting DUTs.

set(dut_type: DutType) → None

```
# SCPI: [SENSe]:CONFigure:MODE:DUT
driver.applications.k30NoiseFigure.sense.configure.mode.dut.set(dut_type =
    enums.DutType.AMPLifier)
```

This command selects the type of DUT you are testing. Note that you have to use [SENSe:]CONFigure:MODE:SYSTem:LO to select if the LO or IF are fixed.

param dut_type

AMPLifier | DDOWnconv | DOWNnconv | UPConv AMPLifier Measurements on fixed frequency DUTs. DOWNnconv Measurements on down-converting DUTs. UPConv Measurements on up-converting DUTs.

6.1.2.9.2.12 System

class SystemCls

System commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.mode.system.clone()
```

Subgroups

6.1.2.9.2.13 Ifreq

class IfreqCls

Ifreq commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.mode.system.ifreq.clone()
```

Subgroups

6.1.2.9.2.14 Frequency

SCPI Commands

```
SENSe:CONFigure:MODE:SYSTem:IF:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:IF:FREQuency
value: float = driver.applications.k30NoiseFigure.sense.configure.mode.system.
↳ ifreq.frequency.get()
```

This command defines the frequency for DUTs with a fixed IF.

return

frequency: Range: 0 Hz to 100 GHz, Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:IF:FREQuency
driver.applications.k30NoiseFigure.sense.configure.mode.system.ifreq.frequency.
↳ set(frequency = 1.0)
```

This command defines the frequency for DUTs with a fixed IF.

param frequency

Range: 0 Hz to 100 GHz, Unit: HZ

6.1.2.9.2.15 Lo

SCPI Commands

```
SENSe:CONFigure:MODE:SYSTem:LO
```

class LoCls

Lo commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → LoType

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:LO
value: enums.LoType = driver.applications.k30NoiseFigure.sense.configure.mode.
↳system.lo.get()
```

This command selects the type of local oscillator you are using. The command is available for measurements on frequency converting DUTs [SENSe:]CONFigure:MODE:DUT .

return

lo_type: FIXed | VARiable FIXed The local oscillator is used as a fixed frequency source. The IF is variable. VARiable The local oscillator is used as a variable frequency source. The IF is fixed.

set(lo_type: LoType) → None

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:LO
driver.applications.k30NoiseFigure.sense.configure.mode.system.lo.set(lo_type =
↳enums.LoType.FIXed)
```

This command selects the type of local oscillator you are using. The command is available for measurements on frequency converting DUTs [SENSe:]CONFigure:MODE:DUT .

param lo_type

FIXed | VARiable FIXed The local oscillator is used as a fixed frequency source. The IF is variable. VARiable The local oscillator is used as a variable frequency source. The IF is fixed.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.configure.mode.system.lo.clone()
```

Subgroups

6.1.2.9.2.16 Frequency

SCPI Commands

```
SENSe:CONFigure:MODE:SYSTem:LO:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:LO:FREQuency
value: float = driver.applications.k30NoiseFigure.sense.configure.mode.system.
↳ lo.frequency.get()
```

This command defines the frequency for DUTs with a fixed LO.

return

lo_frequency: Range: 0 Hz to 100 GHz, Unit: HZ

set(lo_frequency: float) → None

```
# SCPI: [SENSe]:CONFigure:MODE:SYSTem:LO:FREQuency
driver.applications.k30NoiseFigure.sense.configure.mode.system.lo.frequency.
↳ set(lo_frequency = 1.0)
```

This command defines the frequency for DUTs with a fixed LO.

param lo_frequency

Range: 0 Hz to 100 GHz, Unit: HZ

6.1.2.9.2.17 Single

SCPI Commands

```
SENSe:CONFigure:SINgle
```

class SingleCls

Single commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CONFigure:SINgle
driver.applications.k30NoiseFigure.sense.configure.single.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CONFigure:SINgle
driver.applications.k30NoiseFigure.sense.configure.single.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.9.3 Correction

SCPI Commands

```
SENSe:CORRection:RECall
```

class CorrectionCls

Correction commands group definition. 50 total commands, 6 Subgroups, 1 group commands

recall(*recall_file_path: str*) → None

```
# SCPI: [SENSe]:CORRection:RECall
driver.applications.k30NoiseFigure.sense.correction.recall(recall_file_path = '1
→')
```

Sets the calibration results recall filepath and recalls the calibration results.

param recall_file_path

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.clone()
```

Subgroups

6.1.2.9.3.1 Enr

class EnrCls

Enr commands group definition. 23 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.clone()
```

Subgroups

6.1.2.9.3.2 Calibration

class CalibrationCls

Calibration commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.calibration.clone()
```

Subgroups

6.1.2.9.3.3 Mode

SCPI Commands

```
SENSe:CORRection:ENR:CALibration:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMode

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:MODE
value: enums.CorrectionMode = driver.applications.k30NoiseFigure.sense.
↳ correction.enr.calibration.mode.get()
```

This command selects the ENR mode for the calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMon OFF) .

return

mode: SPOT | TABLE SPOT Uses a constant ENR value for all measurement points (see [SENSe:]CORRection:ENR:CALibration:SPOT) . TABLE Uses the contents of the ENR table.

set(mode: CorrectionMode) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:MODE
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.mode.
↳ set(mode = enums.CorrectionMode.SPOT)
```

This command selects the ENR mode for the calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMon OFF) .

param mode

SPOT | TABLE SPOT Uses a constant ENR value for all measurement points (see [SENSe:]CORRection:ENR:CALibration:SPOT) . TABLE Uses the contents of the ENR table.

6.1.2.9.3.4 Sns

class SnsCls

Sns commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.calibration.sns.clone()
```

Subgroups

6.1.2.9.3.5 SrNumber

SCPI Commands

```
SENSe:CORRection:ENR:CALibration:SNS:SRNumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SNS:SRNumber
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.sns.srNumber.get()
```

This command sets and queries the calibration noise source smart noise source serial number.

return

serial_number: No help available

set(serial_number: str) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SNS:SRNumber
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.sns.
↳ srNumber.set(serial_number = '1')
```

This command sets and queries the calibration noise source smart noise source serial number.

param serial_number

No help available

6.1.2.9.3.6 Spot

SCPI Commands

`SENSe:CORRection:ENR:CALibration:SPOT`

class SpotCls

Spot commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.spot.get()
```

This command defines the constant ENR for all measurement points during calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMon OFF) .

return

enr: Range: -999.99 to 999.99, Unit: DB

set(enr: float) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.spot.
↳ set(enr = 1.0)
```

This command defines the constant ENR for all measurement points during calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMon OFF) .

param enr

Range: -999.99 to 999.99, Unit: DB

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.calibration.spot.clone()
```

Subgroups

6.1.2.9.3.7 Cold

SCPI Commands

`SENSe:CORRection:ENR:CALibration:SPOT:COLD`

class ColdCls

Cold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT:COLD
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.spot.cold.get()
```

This command defines a constant temperature of a resistor not supplied with power (Tcold) used during calibration. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE.

return

temperature: Temperature in degrees Kelvin. Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT:COLD
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.spot.cold.
↳ set(temperature = 1.0)
```

This command defines a constant temperature of a resistor not supplied with power (Tcold) used during calibration. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE.

param temperature

Temperature in degrees Kelvin. Unit: K

6.1.2.9.3.8 Hot

SCPI Commands

```
SENSe:CORRection:ENR:CALibration:SPOT:HOT
```

class HotCls

Hot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT:HOT
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.spot.hot.get()
```

This command defines a constant temperature of a resistor supplied with power (Thot) used during calibration. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE.

return

temperature: Temperature in degrees Kelvin. Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:SPOT:HOT
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.spot.hot.
↳ set(temperature = 1.0)
```

This command defines a constant temperature of a resistor supplied with power (Thot) used during calibration. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR:CALibration:TYPE.

param temperature

Temperature in degrees Kelvin. Unit: K

6.1.2.9.3.9 Table

class TableCls

Table commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.calibration.table.
↳ clone()
```

Subgroups

6.1.2.9.3.10 Select

SCPI Commands

```
SENSe:CORRection:ENR:CALibration:TABLE:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:TABLE:SElect
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.table.select.get()
```

This command selects an ENR or temperature table for calibration. Note that the contents of the table are independent of whether you use it for calibration or the actual measurement. When you want to edit a table, regardless if you want to use it later for a measurement or for calibration, you have to use [SENSe:]CORRection:ENR[:MEASurement]:TABLE:SElect. This command only selects a table for calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMON OFF) .

return

table_name: String containing the table name.

set(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:TABLE:SElect
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.table.
↳ select.set(table_name = '1')
```

This command selects an ENR or temperature table for calibration. Note that the contents of the table are independent of whether you use it for calibration or the actual measurement. When you want to edit a table, regardless if you want to use it later for a measurement or for calibration, you have to use [SENSe:]CORRection:ENR[:MEASurement]:TABLE:SElect. This command only selects a table for calibration. This command is available when you use different noise sources for calibration and measurement ([SENSe:]CORRection:ENR:COMMon OFF) .

param table_name

String containing the table name.

6.1.2.9.3.11 TypePy

SCPI Commands

```
SENSe:CORRection:ENR:CALibration:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EnrType

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:TYPE
value: enums.EnrType = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ calibration.typePy.get()
```

This command selects the type of noise source you are using for the calibration.

return

type_py: DIODE Selects a noise source with diode characteristics. RESistor Selects a noise source with resistor characteristics. When you select this noise source type, the application automatically selects the manual measurement mode (see [SENSe:]CONFigure:CONTRol) . SMART Selects a smart noise source.

set(type_py: EnrType) → None

```
# SCPI: [SENSe]:CORRection:ENR:CALibration:TYPE
driver.applications.k30NoiseFigure.sense.correction.enr.calibration.typePy.
↳ set(type_py = enums.EnrType.DIODE)
```

This command selects the type of noise source you are using for the calibration.

param type_py

DIODE Selects a noise source with diode characteristics. RESistor Selects a noise source with resistor characteristics. When you select this noise source type, the application automatically selects the manual measurement mode (see [SENSe:]CONFigure:CONTRol) . SMART Selects a smart noise source.

6.1.2.9.3.12 Common

SCPI Commands

`SENSe:CORRection:ENR:COMMon`

class CommonCls

Common commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:ENR:COMMon
value: bool = driver.applications.k30NoiseFigure.sense.correction.enr.common.
↳get()
```

This command turns the use of a common ENR on or off. For more information see ‘Common Noise Source’.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:ENR:COMMon
driver.applications.k30NoiseFigure.sense.correction.enr.common.set(state =
↳False)
```

This command turns the use of a common ENR on or off. For more information see ‘Common Noise Source’.

param state

ON | OFF | 1 | 0

6.1.2.9.3.13 Measurement

class MeasurementCls

Measurement commands group definition. 13 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.measurement.clone()
```


Subgroups

6.1.2.9.3.14 Mode

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMode

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:MODE
value: enums.CorrectionMode = driver.applications.k30NoiseFigure.sense.
    ↪ correction.enr.measurement.mode.get()
```

This command selects the ENR mode for the actual measurement.

return

mode: SPOT | TABLE SPOT Uses a constant ENR value for all measurement points (see [SENSe:]CORRection:ENR[:MEASurement]:SPOT) . TABLE Uses the contents of the ENR table.

set(mode: CorrectionMode) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:MODE
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.mode.
    ↪ set(mode = enums.CorrectionMode.SPOT)
```

This command selects the ENR mode for the actual measurement.

param mode

SPOT | TABLE SPOT Uses a constant ENR value for all measurement points (see [SENSe:]CORRection:ENR[:MEASurement]:SPOT) . TABLE Uses the contents of the ENR table.

6.1.2.9.3.15 Sns

class SnsCls

Sns commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.measurement.sns.clone()
```

Subgroups

6.1.2.9.3.16 SrNumber

SCPI Commands

SENSe:CORRection:ENR:MEASurement:SNS:SRNumber

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SNS:SRNumber
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.sns.srNumber.get()
```

This command sets and queries the measurement noise source smart noise source serial number.

return

serial_number: No help available

set(serial_number: str) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SNS:SRNumber
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.sns.
↳ srNumber.set(serial_number = '1')
```

This command sets and queries the measurement noise source smart noise source serial number.

param serial_number

No help available

6.1.2.9.3.17 Spot

SCPI Commands

SENSe:CORRection:ENR:MEASurement:SPOT

class SpotCls

Spot commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.spot.get()
```

This command defines the constant ENR for all measurement points during the actual measurement.

return

enr: Unit: DB

set(*enr: float*) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.spot.
↪set(enr = 1.0)
```

This command defines the constant ENR for all measurement points during the actual measurement.

param enr
Unit: DB

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.measurement.spot.clone()
```

Subgroups

6.1.2.9.3.18 Cold

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:SPOT:COLD
```

class ColdCls

Cold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT:COLD
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↪measurement.spot.cold.get()
```

This command defines a constant temperature of a resistor not supplied with power (Tcold) used during measurements. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR[:MEASurement]:TYPE.

return
temperature: Temperature in degrees Kelvin. Unit: K

set(*temperature: float*) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT:COLD
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.spot.cold.
↪set(temperature = 1.0)
```

This command defines a constant temperature of a resistor not supplied with power (Tcold) used during measurements. The command is available when you have selected a noise source with resistor characteristics with [SENSe:]CORRection:ENR[:MEASurement]:TYPE.

param temperature
Temperature in degrees Kelvin. Unit: K

6.1.2.9.3.19 Hot

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:SPOT:HOT
```

class HotCls

Hot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT:HOT
value: float = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.spot.hot.get()
```

This command defines a constant temperature of a resistor supplied with power (Thot) used during measurements. The command is available when you have selected a noise source with resistor characteristics with[SENSe:]CORRection:ENR[:MEASurement]:TYPE .

return

temperature: Temperature in degrees Kelvin. Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:SPOT:HOT
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.spot.hot.
↳ set(temperature = 1.0)
```

This command defines a constant temperature of a resistor supplied with power (Thot) used during measurements. The command is available when you have selected a noise source with resistor characteristics with[SENSe:]CORRection:ENR[:MEASurement]:TYPE .

param temperature

Temperature in degrees Kelvin. Unit: K

6.1.2.9.3.20 Table

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TABLE:DELeTe
```

class TableCls

Table commands group definition. 7 total commands, 4 Subgroups, 1 group commands

delete(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE:DELeTe
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ delete(table_name = '1')
```

This command deletes a temperature table.

param table_name

String containing the name of the table.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ clone()
```

Subgroups

6.1.2.9.3.21 Data

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TABLE:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency_Enr: List[float]: Frequency of the measurement point. Range: 0 Hz to 999.99 GHz, Unit: HZ
- Enr: List[float]: Unit: DB

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE[:DATA]
value: DataStruct = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.table.data.get()
```

This command defines the contents of the currently selected ENR table. Define an ENR for all measurement points. Each entry of the ENR table consists of one measurement point and the corresponding ENR. The individual values are separated by commas or spaces. The table can contain up to 10001 entries. If you create a new table with this command, it overwrites the current entries of the frequency list. To select the ENR table to edit, use [SENSe:]CORRection:ENR[:MEASurement]:TABLE[:DATA].

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency_enr: List[float], enr: List[float]) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE[:DATA]
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.data.
↳ set(frequency_enr = [1.1, 2.2, 3.3], enr = [1.1, 2.2, 3.3])
```

This command defines the contents of the currently selected ENR table. Define an ENR for all measurement points. Each entry of the ENR table consists of one measurement point and the corresponding ENR. The individual values are separated by commas or spaces. The table can contain up to 10001 entries. If you create a new table with this command, it overwrites the current entries of the frequency list. To select the ENR table to edit, use [SENSe:]CORRection:ENR[:MEASurement]:TABLE[:DATA].

param frequency_enr

Frequency of the measurement point. Range: 0 Hz to 999.99 GHz, Unit: HZ

param enr
Unit: DB

6.1.2.9.3.22 ListPy

SCPI Commands

`SENSe:CORRection:ENR:MEASurement:TABLE:LIST`

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE:LIST
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.table.listPy.get()
```

No command help available

return
tables: list

6.1.2.9.3.23 Select

SCPI Commands

`SENSe:CORRection:ENR:MEASurement:TABLE:SElect`

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE:SElect
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.table.select.get()
```

This command selects an ENR or temperature table for the actual measurement. When you want to edit a table, regardless if you want to use it later for a measurement or for calibration, you have to use this command. [SENSe:]CORRection:ENR:CALibration:TABLE:SElect only selects a table for calibration.

return
table_name: No help available

set(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE:SElect
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ select.set(table_name = '1')
```

This command selects an ENR or temperature table for the actual measurement. When you want to edit a table, regardless if you want to use it later for a measurement or for calibration, you have to use this command. [SENSe:]CORRection:ENR:CALibration:TABLE:SElect only selects a table for calibration.

param table_name
No help available

6.1.2.9.3.24 Temperature

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TABLE:TEMPerature:DElete
```

class TemperatureCls

Temperature commands group definition. 3 total commands, 2 Subgroups, 1 group commands

delete(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLE:TEMPerature:DElete
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ temperature.delete(table_name = '1')
```

This command deletes a temperature table.

param table_name
String containing the name of the table.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ temperature.clone()
```

Subgroups

6.1.2.9.3.25 Data

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TABLE:TEMPerature:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: Unit: HZ
- Thot: List[float]: Unit: K
- Tcold: List[float]: Unit: K

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLe:TEMPerature[:DATA]
value: DataStruct = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.table.temperature.data.get()
```

No command help available

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], thot: List[float], tcold: List[float]) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLe:TEMPerature[:DATA]
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.table.
↳ temperature.data.set(frequency = [1.1, 2.2, 3.3], thot = [1.1, 2.2, 3.3],
↳ tcold = [1.1, 2.2, 3.3])
```

No command help available

param frequency

Unit: HZ

param thot

Unit: K

param tcold

Unit: K

6.1.2.9.3.26 ListPy

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TABLe:TEMPerature:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TABLe:TEMPerature:LIST
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.
↳ measurement.table.temperature.listPy.get()
```

This command queries all temperature tables available in the application.

return

tables: list String containing the names of the tables as a comma separated list.

6.1.2.9.3.27 TypePy

SCPI Commands

```
SENSe:CORRection:ENR:MEASurement:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EnrType

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TYPE
value: enums.EnrType = driver.applications.k30NoiseFigure.sense.correction.enr.
    ↳ measurement.typePy.get()
```

This command selects the type of noise source you are using for the measurement.

return

type_py: DIODE Selects a noise source with diode characteristics. RESistor Selects a noise source with resistor characteristics. When you select this noise source type, the application automatically selects the manual measurement mode (see [SENSe:]CONFigure:CONTRol) . SMART Selects a smart noise source.

set(type_py: EnrType) → None

```
# SCPI: [SENSe]:CORRection:ENR[:MEASurement]:TYPE
driver.applications.k30NoiseFigure.sense.correction.enr.measurement.typePy.
    ↳ set(type_py = enums.EnrType.DIODE)
```

This command selects the type of noise source you are using for the measurement.

param type_py

DIODE Selects a noise source with diode characteristics. RESistor Selects a noise source with resistor characteristics. When you select this noise source type, the application automatically selects the manual measurement mode (see [SENSe:]CONFigure:CONTRol) . SMART Selects a smart noise source.

6.1.2.9.3.28 Sns

class SnsCls

Sns commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.sns.clone()
```

Subgroups

6.1.2.9.3.29 Auto

class AutoCls

Auto commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.enr.sns.auto.clone()
```

Subgroups

6.1.2.9.3.30 SrNumber

SCPI Commands

```
SENSe:CORRection:ENR:SNS:AUTO:SRNumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:ENR:SNS:AUTO:SRNumber
value: str = driver.applications.k30NoiseFigure.sense.correction.enr.sns.auto.
↳ srNumber.get()
```

No command help available

```
return
serial_number: No help available
```

6.1.2.9.3.31 State

SCPI Commands

```
SENSe:CORRection:ENR:SNS:AUTO:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:ENR:SNS:AUTO[:STATe]
value: bool = driver.applications.k30NoiseFigure.sense.correction.enr.sns.auto.
↳ state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:ENR:SNS:AUTO[:STATe]
driver.applications.k30NoiseFigure.sense.correction.enr.sns.auto.state.
↪set(state = False)
```

No command help available

param state
No help available

6.1.2.9.3.32 Irejection

SCPI Commands

```
SENSe:CORRection:IREJection
```

class IrejectionCls

Irejection commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:IREJection
value: float = driver.applications.k30NoiseFigure.sense.correction.irejection.
↪get()
```

This command defines the image frequency rejection for the DUT.

return
image_rejection: Range: 0 to 999.99, Unit: DB

set(image_rejection: float) → None

```
# SCPI: [SENSe]:CORRection:IREJection
driver.applications.k30NoiseFigure.sense.correction.irejection.set(image_
↪rejection = 1.0)
```

This command defines the image frequency rejection for the DUT.

param image_rejection
Range: 0 to 999.99, Unit: DB

6.1.2.9.3.33 Loss

class LossCls

Loss commands group definition. 21 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.clone()
```

Subgroups

6.1.2.9.3.34 Calibration

class CalibrationCls

Calibration commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.calibration.clone()
```

Subgroups

6.1.2.9.3.35 Mode

SCPI Commands

```
SENSe:CORRection:LOSS:CALibration:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMode

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:MODE
value: enums.CorrectionMode = driver.applications.k30NoiseFigure.sense.
↳ correction.loss.calibration.mode.get()
```

This command selects the input loss mode.

return

mode: SPOT | TABLE SPOT Uses a constant calibration loss value for all measurement points (see [SENSe:]CORRection:LOSS:CALibration:SPOT) . TABLE Uses the contents of the calibration loss table.

set(mode: CorrectionMode) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:MODE
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.mode.
↳ set(mode = enums.CorrectionMode.SPOT)
```

This command selects the input loss mode.

param mode

SPOT|TABLE SPOT Uses a constant calibration loss value for all measurement points (see [SENSe:]CORRection:LOSS:CALibration:SPOT) . TABLE Uses the contents of the calibration loss table.

6.1.2.9.3.36 Spot**SCPI Commands**

```
SENSe:CORRection:LOSS:CALibration:SPOT
```

class SpotCls

Spot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:SPOT
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ calibration.spot.get()
```

This command defines a constant calibration loss for all measurement points.

return

loss: Range: -999.99 to 999.99, Unit: dB

set(loss: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:SPOT
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.spot.
↳ set(loss = 1.0)
```

This command defines a constant calibration loss for all measurement points.

param loss

Range: -999.99 to 999.99, Unit: dB

6.1.2.9.3.37 Table**SCPI Commands**

```
SENSe:CORRection:LOSS:CALibration:TABLE:DElete
```

class TableCls

Table commands group definition. 4 total commands, 3 Subgroups, 1 group commands

delete(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE:DElete
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.table.
↳ delete(table_name = '1')
```

This command deletes a calibration loss table.

param table_name

String containing the name of the table.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.calibration.table.
↳ clone()
```

Subgroups**6.1.2.9.3.38 Data****SCPI Commands**

```
SENSe:CORRection:LOSS:CALibration:TABLE:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: Frequency of the measurement point. Range: 0 Hz to 999.99 GHz, Unit: HZ
- Loss: List[float]: Loss of the measurement point. Range: -999.99 GHz to 999.99 GHz, Unit: DB

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE[:DATA]
value: DataStruct = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ calibration.table.data.get()
```

This command defines the contents of the currently selected calibration loss table. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. If you create a new table with this command, it overwrites the current entries of the loss table.

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], loss: List[float]) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE[:DATA]
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.table.data.
↳ set(frequency = [1.1, 2.2, 3.3], loss = [1.1, 2.2, 3.3])
```

This command defines the contents of the currently selected calibration loss table. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. If you create a new table with this command, it overwrites the current entries of the loss table.

param frequency

Frequency of the measurement point. Range: 0 Hz to 999.99 GHz, Unit: HZ

param loss

Loss of the measurement point. Range: -999.99 GHz to 999.99 GHz, Unit: DB

6.1.2.9.3.39 ListPy

SCPI Commands

```
SENSe:CORRection:LOSS:CALibration:TABLE:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE:LIST
value: List[str] = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ calibration.table.listPy.get()
```

This command queries all calibration loss tables available in the application.

return
result: No help available

6.1.2.9.3.40 Select

SCPI Commands

```
SENSe:CORRection:LOSS:CALibration:TABLE:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE:SElect
value: str = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ calibration.table.select.get()
```

This command selects a calibration loss table.

return
table_name: String containing the table name.

set(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TABLE:SElect
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.table.
↳ select.set(table_name = '1')
```

This command selects a calibration loss table.

param table_name
String containing the table name.

6.1.2.9.3.41 Temperature

SCPI Commands

SENSe:CORRection:LOSS:CALibration:TEMPerature

class TemperatureCls

Temperature commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TEMPerature
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ calibration.temperature.get()
```

The specified temperature at the time of measurement is considered in the loss calculation.

return
temperature: Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:CALibration:TEMPerature
driver.applications.k30NoiseFigure.sense.correction.loss.calibration.
↳ temperature.set(temperature = 1.0)
```

The specified temperature at the time of measurement is considered in the loss calculation.

param temperature
Unit: K

6.1.2.9.3.42 InputPy

class InputPyCls

InputPy commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.clone()
```

Subgroups

6.1.2.9.3.43 Mode

SCPI Commands

SENSe:CORRection:LOSS:INPut:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMode

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:MODE
value: enums.CorrectionMode = driver.applications.k30NoiseFigure.sense.
↳ correction.loss.inputPy.mode.get()
```

This command selects the input loss mode.

return

mode: SPOT | TABLE SPOT Uses a constant input loss value for all measurement points (see[SENSe:]CORRection:LOSS:INPut:SPOT) . TABLE Uses the contents of the input loss table.

set(mode: CorrectionMode) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:MODE
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.mode.set(mode,
↳ enums.CorrectionMode.SPOT)
```

This command selects the input loss mode.

param mode

SPOT | TABLE SPOT Uses a constant input loss value for all measurement points (see[SENSe:]CORRection:LOSS:INPut:SPOT) . TABLE Uses the contents of the input loss table.

6.1.2.9.3.44 Spot**SCPI Commands**

SENSe:CORRection:LOSS:INPut:SPOT

class SpotCls

Spot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:SPOT
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.
↳ spot.get()
```

This command defines a constant input loss for all measurement points.

return

loss: Range: -999.99 to 999.99, Unit: DB

set(loss: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:SPOT
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.spot.set(loss,
↳ 1.0)
```

This command defines a constant input loss for all measurement points.

param loss

Range: -999.99 to 999.99, Unit: DB

6.1.2.9.3.45 Table**SCPI Commands**

SENSe:CORRection:LOSS:INPut:TABLE:DELeTe

class TableCls

Table commands group definition. 4 total commands, 3 Subgroups, 1 group commands

delete(*table_name: str*) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE:DELeTe
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.table.
↳ delete(table_name = '1')
```

This command deletes an input loss table.

param table_name

String containing the name of the table.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.table.clone()
```

Subgroups**6.1.2.9.3.46 Data****SCPI Commands**

SENSe:CORRection:LOSS:INPut:TABLE:DATA

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: Frequency of the measurement point. Range: 0 dB to 999.99 dB, Unit: HZ
- Loss: List[float]: Loss of the measurement point. Range: -999.99 dB to 999.99 dB, Unit: DB

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE[:DATA]
value: DataStruct = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ inputPy.table.data.get()
```

This command defines the contents of the currently selected input loss table. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. The table should contain an input loss for all measurement points. If you create a new table with this command, it will overwrite the current entries of the loss table.

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], loss: List[float]) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE[:DATA]
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.table.data.
↪set(frequency = [1.1, 2.2, 3.3], loss = [1.1, 2.2, 3.3])
```

This command defines the contents of the currently selected input loss table. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. The table should contain an input loss for all measurement points. If you create a new table with this command, it will overwrite the current entries of the loss table.

param frequency

Frequency of the measurement point. Range: 0 dB to 999.99 dB, Unit: HZ

param loss

Loss of the measurement point. Range: -999.99 dB to 999.99 dB, Unit: DB

6.1.2.9.3.47 ListPy

SCPI Commands

```
SENSe:CORRection:LOSS:INPut:TABLE:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE:LIST
value: List[str] = driver.applications.k30NoiseFigure.sense.correction.loss.
↪inputPy.table.listPy.get()
```

This command queries all input loss tables available in the application.

return

result: No help available

6.1.2.9.3.48 Select

SCPI Commands

```
SENSe:CORRection:LOSS:INPut:TABLE:SELEct
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE:SElect
value: str = driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.
↳ table.select.get()
```

This command selects an input loss table.

return
table_name: String containing the table name.

set(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TABLE:SElect
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.table.select.
↳ set(table_name = '1')
```

This command selects an input loss table.

param table_name
String containing the table name.

6.1.2.9.3.49 Temperature

SCPI Commands

SENSe:CORRection:LOSS:INPut:TEMPerature

class TemperatureCls

Temperature commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TEMPerature
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.
↳ temperature.get()
```

The specified temperature at the time of measurement is considered in the loss calculation.

return
temperature: Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:INPut:TEMPerature
driver.applications.k30NoiseFigure.sense.correction.loss.inputPy.temperature.
↳ set(temperature = 1.0)
```

The specified temperature at the time of measurement is considered in the loss calculation.

param temperature
Unit: K

6.1.2.9.3.50 Output

class OutputCls

Output commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.output.clone()
```

Subgroups

6.1.2.9.3.51 Mode

SCPI Commands

```
SENSe:CORRection:LOSS:OUTPut:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMode

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:MODE
value: enums.CorrectionMode = driver.applications.k30NoiseFigure.sense.
↳correction.loss.output.mode.get()
```

This command selects the output loss mode.

return

mode: SPOT | TABLE SPOT Uses a constant output loss value for all measurement points (see[SENSe:]CORRection:LOSS:OUTPut:SPOT) . TABLE Uses the contents of the output loss table.

set(mode: CorrectionMode) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:MODE
driver.applications.k30NoiseFigure.sense.correction.loss.output.mode.set(mode =
↳enums.CorrectionMode.SPOT)
```

This command selects the output loss mode.

param mode

SPOT | TABLE SPOT Uses a constant output loss value for all measurement points (see[SENSe:]CORRection:LOSS:OUTPut:SPOT) . TABLE Uses the contents of the output loss table.

6.1.2.9.3.52 Spot

SCPI Commands

SENSe:CORRection:LOSS:OUTPut:SPOT

class SpotCls

Spot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:SPOT
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.output.
↳spot.get()
```

This command defines a constant output loss for all measurement points.

return

loss: Range: -999.99 to 999.99, Unit: DB

set(loss: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:SPOT
driver.applications.k30NoiseFigure.sense.correction.loss.output.spot.set(loss =
↳1.0)
```

This command defines a constant output loss for all measurement points.

param loss

Range: -999.99 to 999.99, Unit: DB

6.1.2.9.3.53 Table

SCPI Commands

SENSe:CORRection:LOSS:OUTPut:TABLE:DELeTe

class TableCls

Table commands group definition. 4 total commands, 3 Subgroups, 1 group commands

delete(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE:DELeTe
driver.applications.k30NoiseFigure.sense.correction.loss.output.table.
↳delete(table_name = '1')
```

No command help available

param table_name

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.loss.output.table.clone()
```

Subgroups

6.1.2.9.3.54 Data

SCPI Commands

```
SENSe:CORRection:LOSS:OUTPut:TABLE:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: Frequency of the measurement point. Range: 0 dB to 999.99 dB, Unit: HZ
- Loss: List[float]: Loss of the measurement point. Range: -999.99 dB to 999.99 dB, Unit: DB

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE[:DATA]
value: DataStruct = driver.applications.k30NoiseFigure.sense.correction.loss.
    ↳ output.table.data.get()
```

This command defines the contents of the currently selected output loss table. The table should contain an output loss for all measurement points. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. If you create a new table with this command, it will overwrite the current entries of the frequency list.

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], loss: List[float]) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE[:DATA]
driver.applications.k30NoiseFigure.sense.correction.loss.output.table.data.
    ↳ set(frequency = [1.1, 2.2, 3.3], loss = [1.1, 2.2, 3.3])
```

This command defines the contents of the currently selected output loss table. The table should contain an output loss for all measurement points. Each entry of the loss table consists of one measurement point and the corresponding loss. The table can contain up to 10001 entries. If you create a new table with this command, it will overwrite the current entries of the frequency list.

param frequency

Frequency of the measurement point. Range: 0 dB to 999.99 dB, Unit: HZ

param loss

Loss of the measurement point. Range: -999.99 dB to 999.99 dB, Unit: DB

6.1.2.9.3.55 ListPy

SCPI Commands

`SENSe:CORRection:LOSS:OUTPut:TABLE:LIST`

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE:LIST
value: List[str] = driver.applications.k30NoiseFigure.sense.correction.loss.
↳ output.table.listPy.get()
```

This command queries all output loss tables available in the application.

return
result: No help available

6.1.2.9.3.56 Select

SCPI Commands

`SENSe:CORRection:LOSS:OUTPut:TABLE:SElect`

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE:SElect
value: str = driver.applications.k30NoiseFigure.sense.correction.loss.output.
↳ table.select.get()
```

No command help available

return
table_name: No help available

set(table_name: str) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TABLE:SElect
driver.applications.k30NoiseFigure.sense.correction.loss.output.table.select.
↳ set(table_name = '1')
```

No command help available

param table_name
No help available

6.1.2.9.3.57 Temperature

SCPI Commands

```
SENSe:CORRection:LOSS:OUTPut:TEMPerature
```

class TemperatureCls

Temperature commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TEMPerature
value: float = driver.applications.k30NoiseFigure.sense.correction.loss.output.
↳ temperature.get()
```

The specified temperature at the time of measurement is considered in the loss calculation.

return
temperature: numeric value Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:LOSS:OUTPut:TEMPerature
driver.applications.k30NoiseFigure.sense.correction.loss.output.temperature.
↳ set(temperature = 1.0)
```

The specified temperature at the time of measurement is considered in the loss calculation.

param temperature
numeric value Unit: K

6.1.2.9.3.58 Save

SCPI Commands

```
SENSe:CORRection:SAVE
```

class SaveCls

Save commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:SAVE
value: str = driver.applications.k30NoiseFigure.sense.correction.save.get()
```

Queries and sets the calibration results save filepath and if set saves the calibration results.

return
save_file_path: No help available

set(save_file_path: str) → None

```
# SCPI: [SENSe]:CORRection:SAVE
driver.applications.k30NoiseFigure.sense.correction.save.set(save_file_path = '1
↳ ')
```

Queries and sets the calibration results save filepath and if set saves the calibration results.

param save_file_path
No help available

6.1.2.9.3.59 State

SCPI Commands

SENSe:CORRection:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

<pre># SCPI: [SENSe]:CORRection[:STATe] value: bool = driver.applications.k30NoiseFigure.sense.correction.state.get()</pre>

This command includes or excludes calibration data in the actual measurement (see ‘2nd Stage Correction’ for more information) .

return
state: ON | OFF | 1 | 0

set(state: bool) → None

<pre># SCPI: [SENSe]:CORRection[:STATe] driver.applications.k30NoiseFigure.sense.correction.state.set(state = False)</pre>
--

This command includes or excludes calibration data in the actual measurement (see ‘2nd Stage Correction’ for more information) .

param state
ON | OFF | 1 | 0

6.1.2.9.3.60 Temperature

SCPI Commands

SENSe:CORRection:TEMPerature

class TemperatureCls

Temperature commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

<pre># SCPI: [SENSe]:CORRection:TEMPerature value: float = driver.applications.k30NoiseFigure.sense.correction.temperature. ↪get()</pre>
--

This command defines the room temperature of the measurement environment. The temperature is taken into account when calculating noise results.

return
 temperature: Range: 278.15 to 318.15, Unit: K

set(temperature: float) → None

```
# SCPI: [SENSe]:CORRection:TEMPerature
driver.applications.k30NoiseFigure.sense.correction.temperature.set(temperature_
↪= 1.0)
```

This command defines the room temperature of the measurement environment. The temperature is taken into account when calculating noise results.

param temperature
 Range: 278.15 to 318.15, Unit: K

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.correction.temperature.clone()
```

Subgroups

6.1.2.9.3.61 Control

SCPI Commands

```
SENSe:CORRection:TEMPerature:CONTRol
```

class ControlCls

Control commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:CORRection:TEMPerature:CONTRol
value: enums.AutoManualMode = driver.applications.k30NoiseFigure.sense.
↪correction.temperature.control.get()
```

No command help available

return
 mode: No help available

set(mode: AutoManualMode) → None

```
# SCPI: [SENSe]:CORRection:TEMPerature:CONTRol
driver.applications.k30NoiseFigure.sense.correction.temperature.control.
↪set(mode = enums.AutoManualMode.AUTO)
```

No command help available

param mode
 No help available

6.1.2.9.4 Frequency

class FrequencyCls

Frequency commands group definition. 9 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.frequency.clone()
```

Subgroups

6.1.2.9.4.1 Center

SCPI Commands

```
SENSe:FREQuency:CENTer
```

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:CENTer
value: float = driver.applications.k30NoiseFigure.sense.frequency.center.get()
```

This command defines the center frequency.

return

frequency: The allowed range and fmax is specified in the data sheet. Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:CENTer
driver.applications.k30NoiseFigure.sense.frequency.center.set(frequency = 1.0)
```

This command defines the center frequency.

param frequency

The allowed range and fmax is specified in the data sheet. Unit: Hz

6.1.2.9.4.2 Points

SCPI Commands

```
SENSe:FREQuency:POINts
```

class PointsCls

Points commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:POINts
value: float = driver.applications.k30NoiseFigure.sense.frequency.points.get()
```

This command defines the number of measurement points analyzed during a sweep.

return
sweep_points: Range: 1 to 10001

set(sweep_points: float) → None

```
# SCPI: [SENSe]:FREQuency:POINts
driver.applications.k30NoiseFigure.sense.frequency.points.set(sweep_points = 1.
↪0)
```

This command defines the number of measurement points analyzed during a sweep.

param sweep_points
Range: 1 to 10001

6.1.2.9.4.3 Single

SCPI Commands

```
SENSe:FREQuency:SINGle
```

class SingleCls

Single commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:SINGle
value: float = driver.applications.k30NoiseFigure.sense.frequency.single.get()
```

This command defines the frequency for single frequency measurements.

return
frequency: The minimum and maximum frequency depend on the hardware. Refer to the datasheet for details. Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:SINGle
driver.applications.k30NoiseFigure.sense.frequency.single.set(frequency = 1.0)
```

This command defines the frequency for single frequency measurements.

param frequency
The minimum and maximum frequency depend on the hardware. Refer to the datasheet for details. Unit: HZ

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.frequency.single.clone()
```

Subgroups

6.1.2.9.4.4 Coupled

SCPI Commands

```
SENSe:FREQuency:SINGle:COUPled
```

class CoupledCls

Coupled commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:FREQuency:SINGle:COUPled
value: bool = driver.applications.k30NoiseFigure.sense.frequency.single.coupled.
↳get()
```

Couples or decouples frequency selection to the contents of a sweep list.

return

state: ON | OFF | 0 | 1 OFF | 0 Decouples frequency selection ON | 1 Couples frequency selection

set(state: bool) → None

```
# SCPI: [SENSe]:FREQuency:SINGle:COUPled
driver.applications.k30NoiseFigure.sense.frequency.single.coupled.set(state =
↳False)
```

Couples or decouples frequency selection to the contents of a sweep list.

param state

ON | OFF | 0 | 1 OFF | 0 Decouples frequency selection ON | 1 Couples frequency selection

6.1.2.9.4.5 Span

SCPI Commands

```
SENSe:FREQuency:SPAN
```

class SpanCls

Span commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:SPAN
value: float = driver.applications.k30NoiseFigure.sense.frequency.span.get()
```

This command defines the frequency span. If you change the span, the application creates a new frequency list.

```
return
span: Unit: Hz
```

```
set(span: float) → None
```

```
# SCPI: [SENSe]:FREQuency:SPAN
driver.applications.k30NoiseFigure.sense.frequency.span.set(span = 1.0)
```

This command defines the frequency span. If you change the span, the application creates a new frequency list.

```
param span
Unit: Hz
```

6.1.2.9.4.6 Start

SCPI Commands

```
SENSe:FREQuency:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get() → float
```

```
# SCPI: [SENSe]:FREQuency:STARt
value: float = driver.applications.k30NoiseFigure.sense.frequency.start.get()
```

This command defines the start frequency. If you change the start frequency, the application creates a new frequency list.

```
return
frequency: Unit: HZ
```

```
set(frequency: float) → None
```

```
# SCPI: [SENSe]:FREQuency:STARt
driver.applications.k30NoiseFigure.sense.frequency.start.set(frequency = 1.0)
```

This command defines the start frequency. If you change the start frequency, the application creates a new frequency list.

```
param frequency
Unit: HZ
```

6.1.2.9.4.7 Step

SCPI Commands

SENSe:FREQuency:STEP

class StepCls

Step commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:STEP
value: float = driver.applications.k30NoiseFigure.sense.frequency.step.get()
```

This command defines the frequency stepsize in the frequency table. The stepsize corresponds to the distance from one measurement point to another. If you change the stepsize, the application creates a new frequency list.

return

stepsize: Range: 0 Hz to span, Unit: HZ

set(stepsize: float) → None

```
# SCPI: [SENSe]:FREQuency:STEP
driver.applications.k30NoiseFigure.sense.frequency.step.set(stepsize = 1.0)
```

This command defines the frequency stepsize in the frequency table. The stepsize corresponds to the distance from one measurement point to another. If you change the stepsize, the application creates a new frequency list.

param stepsize

Range: 0 Hz to span, Unit: HZ

6.1.2.9.4.8 Stop

SCPI Commands

SENSe:FREQuency:STOP

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:STOP
value: float = driver.applications.k30NoiseFigure.sense.frequency.stop.get()
```

This command defines the stop frequency. If you change the stop frequency, the application creates a new frequency list.

return

frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQUENCY:STOP
driver.applications.k30NoiseFigure.sense.frequency.stop.set(frequency = 1.0)
```

This command defines the stop frequency. If you change the stop frequency, the application creates a new frequency list.

param frequency
Unit: HZ

6.1.2.9.4.9 Table

class TableCls

Table commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.frequency.table.clone()
```

Subgroups

6.1.2.9.4.10 Data

SCPI Commands

```
SENSe:FREQUENCY:TABLE:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:FREQUENCY:TABLE:DATA
value: List[float] = driver.applications.k30NoiseFigure.sense.frequency.table.
↳data.get()
```

This command defines the contents of the frequency table. The command overwrites the current contents of the frequency table.

return
frequency: Defines a frequency for each entry in the frequency table. A frequency table can contain up to 10001 entries. Range: 0 Hz to fmax, Unit: HZ

set(frequency: List[float]) → None

```
# SCPI: [SENSe]:FREQUENCY:TABLE:DATA
driver.applications.k30NoiseFigure.sense.frequency.table.data.set(frequency =
↳[1.1, 2.2, 3.3])
```

This command defines the contents of the frequency table. The command overwrites the current contents of the frequency table.

param frequency

Defines a frequency for each entry in the frequency table. A frequency table can contain up to 10001 entries. Range: 0 Hz to fmax, Unit: HZ

6.1.2.9.5 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k30NoiseFigure.sense.probe.repcap_probe_get()
driver.applications.k30NoiseFigure.sense.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.probe.clone()
```

Subgroups

6.1.2.9.5.1 Id

class IdCls

Id commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.probe.id.clone()
```

Subgroups

6.1.2.9.5.2 PartNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:PARTnumber
```

class PartNumberCls

PartNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → str

```
# SCPI: [SENSe]:PROBe<pb>:ID:PARTnumber
value: str = driver.applications.k30NoiseFigure.sense.probe.id.partNumber.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

result: No help available

6.1.2.9.5.3 SrNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:SRNumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → str

```
# SCPI: [SENSe]:PROBe<pb>:ID:SRNumber
value: str = driver.applications.k30NoiseFigure.sense.probe.id.srNumber.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

result: No help available

6.1.2.9.6 Sweep

class SweepCls

Sweep commands group definition. 13 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.clone()
```

Subgroups

6.1.2.9.6.1 Count

SCPI Commands

```
SENSe:SWEEP:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEP:COUNT
value: float = driver.applications.k30NoiseFigure.sense.sweep.count.get()
```

This command defines the number of measurements that are used to average the results.

return

averages: Number of measurements that are performed at a single frequency before average results are displayed. If you set an average of 0 or 1, the application performs a single measurement at each frequency. Range: 0 to 32767

set(averages: float) → None

```
# SCPI: [SENSe]:SWEEP:COUNT
driver.applications.k30NoiseFigure.sense.sweep.count.set(averages = 1.0)
```

This command defines the number of measurements that are used to average the results.

param averages

Number of measurements that are performed at a single frequency before average results are displayed. If you set an average of 0 or 1, the application performs a single measurement at each frequency. Range: 0 to 32767

6.1.2.9.6.2 Egate

SCPI Commands

```
SENSe:SWEEP:EGATE
```

class EgateCls

Egate commands group definition. 11 total commands, 8 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEep:EGATe
value: bool = driver.applications.k30NoiseFigure.sense.sweep.egate.get()
```

This command turns gated measurements on and off. See ‘(Measurement) Points’.

```
return
    state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function
    on
```

set(state: bool) → None

```
# SCPI: [SENSe]:SWEep:EGATe
driver.applications.k30NoiseFigure.sense.sweep.egate.set(state = False)
```

This command turns gated measurements on and off. See ‘(Measurement) Points’.

```
param state
    ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.egate.clone()
```

Subgroups

6.1.2.9.6.3 Auto

SCPI Commands

```
SENSe:SWEep:EGATe:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEep:EGATe:AUTO
value: bool = driver.applications.k30NoiseFigure.sense.sweep.egate.auto.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: [SENSe]:SWEep:EGATe:AUTO
driver.applications.k30NoiseFigure.sense.sweep.egate.auto.set(state = False)
```

No command help available

```
param state
    No help available
```

6.1.2.9.6.4 Continuous

class ContinuousCls

Continuous commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.clone()
```

Subgroups

6.1.2.9.6.5 Pcount

SCPI Commands

```
SENSe:SWEep:EGATe:CONTinuous:PCount
```

class PcountCls

Pcount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:CONTinuous:PCount
value: float = driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.
↳pcount.get()
```

Defines the number of gate periods to be measured after a single trigger event.

return

amount: integer Range: 1 to 65535

set(amount: float) → None

```
# SCPI: [SENSe]:SWEep:EGATe:CONTinuous:PCount
driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.pcount.
↳set(amount = 1.0)
```

Defines the number of gate periods to be measured after a single trigger event.

param amount

integer Range: 1 to 65535

6.1.2.9.6.6 Plength

SCPI Commands

```
SENSe:SWEEp:EGATe:CONTInuous:PLENgtH
```

class PlengthCls

Plength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEp:EGATe:CONTInuous:PLENgtH
value: float = driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.
    ↪ plength.get()
```

Defines the length in seconds of a single gate period in continuous gating. The length is determined from the beginning of one gate measurement to the beginning of the next one.

return
time: Range: 125 ns to 30 s, Unit: S

set(time: float) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:CONTInuous:PLENgtH
driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.plength.
    ↪ set(time = 1.0)
```

Defines the length in seconds of a single gate period in continuous gating. The length is determined from the beginning of one gate measurement to the beginning of the next one.

param time
Range: 125 ns to 30 s, Unit: S

6.1.2.9.6.7 State

SCPI Commands

```
SENSe:SWEEp:EGATe:CONTInuous:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEEp:EGATe:CONTInuous[:STATe]
value: bool = driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.
    ↪ state.get()
```

Activates or deactivates continuous gating. This setting is only available if [SENSe:]SWEEp:EGATe is 'On'.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool*) → None

```
# SCPI: [SENSe]:SWEep:EGATe:CONTInuous[:STATe]
driver.applications.k30NoiseFigure.sense.sweep.egate.continuous.state.set(state_
↪ False)
```

Activates or deactivates continuous gating. This setting is only available if [SENSe:]SWEep:EGATe is 'On'.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.2.9.6.8 Holdoff

SCPI Commands

```
SENSe:SWEep:EGATe:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:HOLDoff
value: float = driver.applications.k30NoiseFigure.sense.sweep.egate.holdoff.
↪ get()
```

This command defines the delay time between the gate signal and the continuation of the measurement.

return

delay_time: Range: 0 s to 30 s, Unit: S

set(*delay_time: float*) → None

```
# SCPI: [SENSe]:SWEep:EGATe:HOLDoff
driver.applications.k30NoiseFigure.sense.sweep.egate.holdoff.set(delay_time = 1.
↪ 0)
```

This command defines the delay time between the gate signal and the continuation of the measurement.

param delay_time

Range: 0 s to 30 s, Unit: S

6.1.2.9.6.9 Length

SCPI Commands

```
SENSe:SWEep:EGATe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEP:EGATE:LENGth
value: float = driver.applications.k30NoiseFigure.sense.sweep.egate.length.get()
```

This command defines the gate length.

```
return
    gate_length: Range: 125 ns to 30 s, Unit: S
```

set(gate_length: float) → None

```
# SCPI: [SENSe]:SWEEP:EGATE:LENGth
driver.applications.k30NoiseFigure.sense.sweep.egate.length.set(gate_length = 1.
↪0)
```

This command defines the gate length.

```
param gate_length
    Range: 125 ns to 30 s, Unit: S
```

6.1.2.9.6.10 Level

class LevelCls

Level commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.egate.level.clone()
```

Subgroups

6.1.2.9.6.11 External<ExternalPort>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.applications.k30NoiseFigure.sense.sweep.egate.level.external.repcap_
↪externalPort_get()
driver.applications.k30NoiseFigure.sense.sweep.egate.level.external.repcap_externalPort_
↪set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
SENSe:SWEEp:EGATe:LEVel:EXTeRnal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(*externalPort*=*ExternalPort.Default*) → float

```
# SCPI: [SENSe]:SWEEp:EGATe:LEVel[:EXTeRnal<1|2|3>]
value: float = driver.applications.k30NoiseFigure.sense.sweep.egate.level.
↪external.get(externalPort = repcap.ExternalPort.Default)
```

No command help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

level: No help available

set(*level*: float, *externalPort*=*ExternalPort.Default*) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:LEVel[:EXTeRnal<1|2|3>]
driver.applications.k30NoiseFigure.sense.sweep.egate.level.external.set(level = ↪
↪1.0, externalPort = repcap.ExternalPort.Default)
```

No command help available

param level

No help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.egate.level.external.clone()
```

6.1.2.9.6.12 Polarity

SCPI Commands

```
SENSe:SWEEp:EGATe:POLarity
```

class PolarityCls

Polarity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SlopeType

```
# SCPI: [SENSe]:SWEep:EGATe:POLarity
value: enums.SlopeType = driver.applications.k30NoiseFigure.sense.sweep.egate.
↳polarity.get()
```

This command selects the polarity of an external gate signal. The setting applies both to the edge of an edge-triggered signal and the level of a level-triggered signal.

return
polarity: POSitive | NEGative

set(polarity: SlopeType) → None

```
# SCPI: [SENSe]:SWEep:EGATe:POLarity
driver.applications.k30NoiseFigure.sense.sweep.egate.polarity.set(polarity =
↳enums.SlopeType.NEGative)
```

This command selects the polarity of an external gate signal. The setting applies both to the edge of an edge-triggered signal and the level of a level-triggered signal.

param polarity
POSitive | NEGative

6.1.2.9.6.13 Source

SCPI Commands

SENSe:SWEep:EGATe:SOURce

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → GatedSourceK30

```
# SCPI: [SENSe]:SWEep:EGATe:SOURce
value: enums.GatedSourceK30 = driver.applications.k30NoiseFigure.sense.sweep.
↳egate.source.get()
```

This command selects the signal source for gated measurements. If an IF power signal is used, the gate is opened as soon as a signal at > -20 dBm is detected within the IF path bandwidth (10 MHz) . For more information see ‘Trigger Source’.

return
source: EXTernal | EXT2 | EXT3 | IFPower | IQPower | VIDeo | RFPower | PSEN

set(source: GatedSourceK30) → None

```
# SCPI: [SENSe]:SWEep:EGATe:SOURce
driver.applications.k30NoiseFigure.sense.sweep.egate.source.set(source = enums.
↳GatedSourceK30.EXT2)
```

This command selects the signal source for gated measurements. If an IF power signal is used, the gate is opened as soon as a signal at > -20 dBm is detected within the IF path bandwidth (10 MHz) . For more information see ‘Trigger Source’.

param source

EXTernal | EXT2 | EXT3 | IFPower | IQPower | VIDEo | RFPower | PSEN

6.1.2.9.6.14 TypePy**SCPI Commands**

SENSe:SWEEp:EGATe:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EgateType

```
# SCPI: [SENSe]:SWEEp:EGATe:TYPE
value: enums.EgateType = driver.applications.k30NoiseFigure.sense.sweep.egate.
↳ typePy.get()
```

This command selects the way gated measurements are triggered.

return

type_py: LEVel The trigger event for the gate to open is a particular power level. After the gate signal has been detected, the gate remains open until the signal disappears. EDGE The trigger event for the gate to open is the detection of the signal edge. After the gate signal has been detected, the gate remains open until the gate length is over.

set(type_py: EgateType) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:TYPE
driver.applications.k30NoiseFigure.sense.sweep.egate.typePy.set(type_py = enums.
↳ EgateType.EDGE)
```

This command selects the way gated measurements are triggered.

param type_py

LEVel The trigger event for the gate to open is a particular power level. After the gate signal has been detected, the gate remains open until the signal disappears. EDGE The trigger event for the gate to open is the detection of the signal edge. After the gate signal has been detected, the gate remains open until the gate length is over.

6.1.2.9.6.15 Time**class TimeCls**

Time commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.sense.sweep.time.clone()
```

Subgroups

6.1.2.9.6.16 Auto

SCPI Commands

```
SENSe:SWEp:TIME:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEp:TIME:AUTO
value: bool = driver.applications.k30NoiseFigure.sense.sweep.time.auto.get()
```

If enabled, the sweep time is automatically selected, depending on the current frequency of the sweep point, as defined in the frequency table (see ‘Using a frequency table’) . If disabled, the value defined by [SENSe:]SWEp:TIME is used.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:SWEp:TIME:AUTO
driver.applications.k30NoiseFigure.sense.sweep.time.auto.set(state = False)
```

If enabled, the sweep time is automatically selected, depending on the current frequency of the sweep point, as defined in the frequency table (see ‘Using a frequency table’) . If disabled, the value defined by [SENSe:]SWEp:TIME is used.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.2.10 Source

class SourceCls

Source commands group definition. 38 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.clone()
```

Subgroups

6.1.2.10.1 Current

class CurrentCls

Current commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.clone()
```

Subgroups

6.1.2.10.1.1 Auxiliary

class AuxiliaryCls

Auxiliary commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.auxiliary.clone()
```

Subgroups

6.1.2.10.1.2 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.auxiliary.limit.clone()
```

Subgroups

6.1.2.10.1.3 High

SCPI Commands

```
SOURce:CURRent:AUX:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:CURRent:AUX:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.current.auxiliary.
↳limit.high.get()
```

No command help available

return
current: No help available

6.1.2.10.1.4 Control<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.applications.k30NoiseFigure.source.current.control.repcap_source_get()
driver.applications.k30NoiseFigure.source.current.control.repcap_source_set(repcap.
↳Source.Nr1)
```

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.control.clone()
```

Subgroups

6.1.2.10.1.5 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.control.limit.clone()
```

Subgroups

6.1.2.10.1.6 High

SCPI Commands

```
SOURce:CURRent:CONTRol<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:CURRent:CONTRol<i>:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.current.control.limit.
↳high.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

current: No help available

6.1.2.10.1.7 Power<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.applications.k30NoiseFigure.source.current.power.repcap_source_get()
driver.applications.k30NoiseFigure.source.current.power.repcap_source_set(repcap.Source.
↳Nr1)
```

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.power.clone()
```

Subgroups

6.1.2.10.1.8 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.power.limit.clone()
```

Subgroups

6.1.2.10.1.9 High

SCPI Commands

```
SOURce:CURRent:POWer<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:CURRent:POWer<i>:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.current.power.limit.
↳ high.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

current: No help available

set(current: float, source=Source.Default) → None

```
# SCPI: SOURce:CURRent:POWer<i>:LIMit:HIGH
driver.applications.k30NoiseFigure.source.current.power.limit.high.set(current.
↳ = 1.0, source = repcap.Source.Default)
```

No command help available

param current

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.1.10 Sequence**class SequenceCls**

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.current.sequence.clone()
```

Subgroups**6.1.2.10.1.11 Result****SCPI Commands**

```
SOURce:CURRent:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:CURRent:SEquence:RESult
value: float = driver.applications.k30NoiseFigure.source.current.sequence.
↳ result.get()
```

No command help available

return

result: No help available

6.1.2.10.2 External**class ExternalCls**

External commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.clone()
```

Subgroups

6.1.2.10.2.1 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.frequency.clone()
```

Subgroups

6.1.2.10.2.2 Factor

class FactorCls

Factor commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.frequency.factor.clone()
```

Subgroups

6.1.2.10.2.3 Denominator

SCPI Commands

```
SOURce:EXternal:FREquency:FACTOR:DENominator
```

class DenominatorCls

Denominator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:EXternal:FREquency[:FACTOR]:DENominator
value: float = driver.applications.k30NoiseFigure.source.external.frequency.
    ↪ factor.denominator.get()
```

No command help available

```
return
    denominator: Unit: HZ
```

set(*denominator: float*) → None

```
# SCPI: SOURce:EXtErnal:FREQuency[:FACTOR]:DENominator
driver.applications.k30NoiseFigure.source.external.frequency.factor.denominator.
↪set(denominator = 1.0)
```

No command help available

```
param denominator
    Unit: HZ
```

6.1.2.10.2.4 Numerator

SCPI Commands

```
SOURce:EXtErnal:FREQuency:FACTOR:NUMerator
```

class NumeratorCls

Numerator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:EXtErnal:FREQuency[:FACTOR]:NUMerator
value: float = driver.applications.k30NoiseFigure.source.external.frequency.
↪factor.numerator.get()
```

No command help available

```
return
    numerator: No help available
```

set(*numerator: float*) → None

```
# SCPI: SOURce:EXtErnal:FREQuency[:FACTOR]:NUMerator
driver.applications.k30NoiseFigure.source.external.frequency.factor.numerator.
↪set(numerator = 1.0)
```

No command help available

```
param numerator
    No help available
```

6.1.2.10.2.5 Offset<FreqOffset>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k30NoiseFigure.source.external.frequency.offset.repcap_
    ↪ freqOffset_get()
driver.applications.k30NoiseFigure.source.external.frequency.offset.repcap_freqOffset_
    ↪ set(repcap.FreqOffset.Nr1)
```

SCPI Commands

```
SOURce:EXTernal:FREquency:OFFSet<FreqOffset>
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: FreqOffset, default value after init: FreqOffset.Nr1

get(freqOffset=FreqOffset.Default) → float

```
# SCPI: SOURce:EXTernal:FREquency:OFFSet<of>
value: float = driver.applications.k30NoiseFigure.source.external.frequency.
    ↪ offset.get(freqOffset = repcap.FreqOffset.Default)
```

No command help available

param freqOffset

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Offset')

return

denominator: Unit: HZ

set(denominator: float, freqOffset=FreqOffset.Default) → None

```
# SCPI: SOURce:EXTernal:FREquency:OFFSet<of>
driver.applications.k30NoiseFigure.source.external.frequency.offset.
    ↪ set(denominator = 1.0, freqOffset = repcap.FreqOffset.Default)
```

No command help available

param denominator

Unit: HZ

param freqOffset

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Offset')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.frequency.offset.clone()
```

6.1.2.10.2.6 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.power.clone()
```

Subgroups

6.1.2.10.2.7 Level

SCPI Commands

```
SOURce:EXternal:POWer:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:EXternal:POWer[:LEVel]
value: float = driver.applications.k30NoiseFigure.source.external.power.level.
↳get()
```

This command sets the output power of the selected generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed and active.

return

level: numeric value Unit: DBM

set(level: float) → None

```
# SCPI: SOURce:EXternal:POWer[:LEVel]
driver.applications.k30NoiseFigure.source.external.power.level.set(level = 1.0)
```

This command sets the output power of the selected generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed and active.

param level

numeric value Unit: DBM

6.1.2.10.2.8 Roscillator

class RoscillatorCls

Roscillator commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.external.roscillator.clone()
```

Subgroups

6.1.2.10.2.9 Source

SCPI Commands

```
SOURce:EXternal:ROSCillator:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SourceInt

```
# SCPI: SOURce:EXternal:ROSCillator[:SOURce]
value: enums.SourceInt = driver.applications.k30NoiseFigure.source.external.
    ↪ roscillator.source.get()
```

This command controls selection of the reference oscillator for the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed. If the external reference oscillator is selected, the reference signal must be connected to the rear panel of the instrument.

return

source: INTERNAL Uses the internal reference. EXTERNAL Uses the external reference; if none is available, an error flag is displayed in the status bar.

set(source: SourceInt) → None

```
# SCPI: SOURce:EXternal:ROSCillator[:SOURce]
driver.applications.k30NoiseFigure.source.external.roscillator.source.
    ↪ set(source = enums.SourceInt.EXTERNAL)
```

This command controls selection of the reference oscillator for the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed. If the external reference oscillator is selected, the reference signal must be connected to the rear panel of the instrument.

param source

INTERNAL Uses the internal reference. EXTERNAL Uses the external reference; if none is available, an error flag is displayed in the status bar.

6.1.2.10.3 Generator

class GeneratorCls

Generator commands group definition. 9 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.generator.clone()
```

Subgroups

6.1.2.10.3.1 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.generator.channel.clone()
```

Subgroups

6.1.2.10.3.2 Coupling

SCPI Commands

```
SOURce:GENerator:CHANnel:COUpling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:CHANnel:COUpling
value: bool = driver.applications.k30NoiseFigure.source.generator.channel.
↳coupling.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SOURce:GENerator:CHANnel:COUpling
driver.applications.k30NoiseFigure.source.generator.channel.coupling.set(state,
↳False)
```


No command help available

param state

No help available

6.1.2.10.3.3 DutBypass

SCPI Commands

SOURce:GENerator:DUTBypass

class DutBypassCls

DutBypass commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:DUTBypass
value: bool = driver.applications.k30NoiseFigure.source.generator.dutBypass.
↳get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SOURce:GENerator:DUTBypass
driver.applications.k30NoiseFigure.source.generator.dutBypass.set(state = False)
```

No command help available

param state

No help available

6.1.2.10.3.4 Frequency

SCPI Commands

SOURce:GENerator:FREQUENCY

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:FREQUENCY
value: float = driver.applications.k30NoiseFigure.source.generator.frequency.
↳get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: SOURce:GENerator:FREquency
driver.applications.k30NoiseFigure.source.generator.frequency.set(frequency = 1.
↪0)
```

No command help available

param frequency
No help available

6.1.2.10.3.5 Level

SCPI Commands

SOURce:GENerator:LEVel

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:LEVel
value: float = driver.applications.k30NoiseFigure.source.generator.level.get()
```

No command help available

return
level: No help available

set(level: float) → None

```
# SCPI: SOURce:GENerator:LEVel
driver.applications.k30NoiseFigure.source.generator.level.set(level = 1.0)
```

No command help available

param level
No help available

6.1.2.10.3.6 Modulation

SCPI Commands

SOURce:GENerator:MODulation

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:MODulation
value: bool = driver.applications.k30NoiseFigure.source.generator.modulation.
↪get()
```

No command help available

```

return
    state: No help available

```

set(state: bool) → None

```

# SCPI: SOURce:GENerator:MODulation
driver.applications.k30NoiseFigure.source.generator.modulation.set(state =
↪False)

```

No command help available

```

param state
    No help available

```

6.1.2.10.3.7 Pulse

class PulseCls

Pulse commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```

# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.generator.pulse.clone()

```

Subgroups

6.1.2.10.3.8 Period

SCPI Commands

```
SOURce:GENerator:PULSe:PERiod
```

class PeriodCls

Period commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```

# SCPI: SOURce:GENerator:PULSe:PERiod
value: float = driver.applications.k30NoiseFigure.source.generator.pulse.period.
↪get()

```

No command help available

```

return
    pulse_period: No help available

```

set(pulse_period: float) → None

```

# SCPI: SOURce:GENerator:PULSe:PERiod
driver.applications.k30NoiseFigure.source.generator.pulse.period.set(pulse_
↪period = 1.0)

```

No command help available

param pulse_period

No help available

6.1.2.10.3.9 Trigger

class TriggerCls

Trigger commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.generator.pulse.trigger.clone()
```

Subgroups

6.1.2.10.3.10 Output

SCPI Commands

```
SOURce:GENerator:PULSe:TRIGger:OUTPut
```

class OutputCls

Output commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SignalLevel

```
# SCPI: SOURce:GENerator:PULSe:TRIGger:OUTPut
value: enums.SignalLevel = driver.applications.k30NoiseFigure.source.generator.
    ↪ pulse.trigger.output.get()
```

No command help available

return

signal_level: No help available

set(signal_level: SignalLevel) → None

```
# SCPI: SOURce:GENerator:PULSe:TRIGger:OUTPut
driver.applications.k30NoiseFigure.source.generator.pulse.trigger.output.
    ↪ set(signal_level = enums.SignalLevel.HIGH)
```

No command help available

param signal_level

No help available

6.1.2.10.3.11 Width

SCPI Commands

SOURce:GENerator:PULSe:WIDTh

class WidthCls

Width commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:PULSe:WIDTh
value: float = driver.applications.k30NoiseFigure.source.generator.pulse.width.
↳get()
```

No command help available

```
return
    pulse_width: No help available
```

set(pulse_width: float) → None

```
# SCPI: SOURce:GENerator:PULSe:WIDTh
driver.applications.k30NoiseFigure.source.generator.pulse.width.set(pulse_width.
↳= 1.0)
```

No command help available

```
param pulse_width
    No help available
```

6.1.2.10.3.12 State

SCPI Commands

SOURce:GENerator:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.generator.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: SOURce:GENerator[:STATe]
driver.applications.k30NoiseFigure.source.generator.state.set(state = False)
```

No command help available

param state

No help available

6.1.2.10.4 Nsource

class NsourceCls

Nsource commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.nsource.clone()
```

Subgroups

6.1.2.10.4.1 State

SCPI Commands

```
SOURce:NSource:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:NSource[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.nsource.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SOURce:NSource[:STATe]
driver.applications.k30NoiseFigure.source.nsource.state.set(state = False)
```

No command help available

param state

No help available

6.1.2.10.5 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.power.clone()
```

Subgroups

6.1.2.10.5.1 Sequence

class SequenceCls

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.power.sequence.clone()
```

Subgroups

6.1.2.10.5.2 Result

SCPI Commands

```
SOURce:POWer:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:POWer:SEquence:RESult
value: float = driver.applications.k30NoiseFigure.source.power.sequence.result.
    ↪get()
```

No command help available

```
return
    result: No help available
```

6.1.2.10.6 Voltage

class VoltageCls

Voltage commands group definition. 18 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.clone()
```

Subgroups

6.1.2.10.6.1 Auxiliary

class AuxiliaryCls

Auxiliary commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.auxiliary.clone()
```

Subgroups

6.1.2.10.6.2 Level

class LevelCls

Level commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.clone()
```

Subgroups

6.1.2.10.6.3 Amplitude

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:AMPLitude
value: float = driver.applications.k30NoiseFigure.source.voltage.auxiliary.
↪level.amplitude.get()
```

No command help available

return

voltage: No help available

set(voltage: float) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:AMPLitude
driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.amplitude.
↪set(voltage = 1.0)
```

No command help available

param voltage

No help available

6.1.2.10.6.4 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.limit.clone()
```

Subgroups

6.1.2.10.6.5 High

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.voltage.auxiliary.
↪level.limit.high.get()
```

No command help available

return

voltage: No help available

set(*voltage: float*) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.limit.high.
↪set(voltage = 1.0)
```

No command help available

param voltage

No help available

6.1.2.10.6.6 Low

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:LOW
value: float = driver.applications.k30NoiseFigure.source.voltage.auxiliary.
↪level.limit.low.get()
```

No command help available

return

voltage: No help available

set(*voltage: float*) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:LOW
driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.limit.low.
↪set(voltage = 1.0)
```

No command help available

param voltage

No help available

6.1.2.10.6.7 State

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:VOLTage:AUX:LEVel[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.
↳ state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel[:STATe]
driver.applications.k30NoiseFigure.source.voltage.auxiliary.level.state.
↳ set(state = False)
```

No command help available

param state
No help available

6.1.2.10.6.8 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.channel.clone()
```

Subgroups

6.1.2.10.6.9 Coupling

SCPI Commands

```
SOURce:VOLTage:CHANnel:COUpling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:VOLTage:CHANnel:COUpling
value: bool = driver.applications.k30NoiseFigure.source.voltage.channel.
↳ coupling.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SOURce:VOLTage:CHANnel:COUpling
driver.applications.k30NoiseFigure.source.voltage.channel.coupling.set(state = ↪
↪False)
```

No command help available

param state

No help available

6.1.2.10.6.10 Control<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.applications.k30NoiseFigure.source.voltage.control.repcap_source_get()
driver.applications.k30NoiseFigure.source.voltage.control.repcap_source_set(repcap.
↪Source.Nr1)
```

class ControlCls

Control commands group definition. 4 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.control.clone()
```

Subgroups

6.1.2.10.6.11 Level

class LevelCls

Level commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.control.level.clone()
```

Subgroups

6.1.2.10.6.12 Amplitude

SCPI Commands

```
SOURce:VOLTage:CONTRol<Source>:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:AMPLitude
value: float = driver.applications.k30NoiseFigure.source.voltage.control.level.
↳ amplitude.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

voltage: No help available

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:AMPLitude
driver.applications.k30NoiseFigure.source.voltage.control.level.amplitude.
↳ set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.1.2.10.6.13 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.control.level.limit.clone()
```

Subgroups

6.1.2.10.6.14 High

SCPI Commands

```
SOURce:VOLTage:CONTRol<Source>:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.voltage.control.level.
↳limit.high.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

voltage: No help available

set(voltage: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:LIMit:HIGH
driver.applications.k30NoiseFigure.source.voltage.control.level.limit.high.
↳set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.1.2.10.6.15 Low

SCPI Commands

```
SOURce:VOLTage:CONTRol<Source>:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:LIMit:LOW
value: float = driver.applications.k30NoiseFigure.source.voltage.control.level.
↳ limit.low.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

voltage: No help available

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel:LIMit:LOW
driver.applications.k30NoiseFigure.source.voltage.control.level.limit.low.
↳ set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.1.2.10.6.16 State

SCPI Commands

```
SOURce:VOLTage:CONTRol<Source>:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → bool

```
# SCPI: SOURce:VOLTage:CONTRol<i>:LEVel[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.voltage.control.level.
↳ state.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

state: No help available

set(state: bool, source=Source.Default) → None

```
# SCPI: SOURCE:VOLTage:CONTrol<i>:LEVel[:STATe]
driver.applications.k30NoiseFigure.source.voltage.control.level.state.set(state_
↪= False, source = repcap.Source.Default)
```

No command help available

param state

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.1.2.10.6.17 Power<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.applications.k30NoiseFigure.source.voltage.power.repcap_source_get()
driver.applications.k30NoiseFigure.source.voltage.power.repcap_source_set(repcap.Source.
↪Nr1)
```

class PowerCls

Power commands group definition. 6 total commands, 2 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.power.clone()
```

Subgroups

6.1.2.10.6.18 Level

class LevelCls

Level commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.power.level.clone()
```

Subgroups

6.1.2.10.6.19 Amplitude

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:AMPLitude
value: float = driver.applications.k30NoiseFigure.source.voltage.power.level.
↳amplitude.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage: No help available

set(voltage: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:AMPLitude
driver.applications.k30NoiseFigure.source.voltage.power.level.amplitude.
↳set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.20 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.power.level.limit.clone()
```

Subgroups

6.1.2.10.6.21 High

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.voltage.power.level.
↳ limit.high.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage: No help available

set(voltage: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:LIMit:HIGH
driver.applications.k30NoiseFigure.source.voltage.power.level.limit.high.
↳ set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.22 Low

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:LIMit:LOW
value: float = driver.applications.k30NoiseFigure.source.voltage.power.level.
↳ limit.low.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage: No help available

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:LIMit:LOW
driver.applications.k30NoiseFigure.source.voltage.power.level.limit.low.
↳ set(voltage = 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.23 Mode

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → PwrLevelMode

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:MODE
value: enums.PwrLevelMode = driver.applications.k30NoiseFigure.source.voltage.
↳ power.level.mode.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

mode: No help available

set(*mode: PwrLevelMode, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel:MODE
driver.applications.k30NoiseFigure.source.voltage.power.level.mode.set(mode =
↳enums.PwrLevelMode.CURRENT, source = repcap.Source.Default)
```

No command help available

param mode

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.24 State

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → bool

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.voltage.power.level.
↳state.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

state: No help available

set(*state: bool, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LEVel[:STATe]
driver.applications.k30NoiseFigure.source.voltage.power.level.state.set(state =
↳False, source = repcap.Source.Default)
```

No command help available

param state

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.25 Limit**class LimitCls**

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.power.limit.clone()
```

Subgroups**6.1.2.10.6.26 High****SCPI Commands**

```
SOURce:VOLTage:POWer<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<i>:LIMit:HIGH
value: float = driver.applications.k30NoiseFigure.source.voltage.power.limit.
↳high.get(source = repcap.Source.Default)
```

No command help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage: No help available

set(voltage: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<i>:LIMit:HIGH
driver.applications.k30NoiseFigure.source.voltage.power.limit.high.set(voltage_
↳= 1.0, source = repcap.Source.Default)
```

No command help available

param voltage

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.1.2.10.6.27 Sequence**class SequenceCls**

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.source.voltage.sequence.clone()
```

Subgroups**6.1.2.10.6.28 Result****SCPI Commands**

```
SOURce:VOLTage:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:SEquence:RESult
value: float = driver.applications.k30NoiseFigure.source.voltage.sequence.
↳ result.get()
```

No command help available

return

result: No help available

6.1.2.10.6.29 State**SCPI Commands**

```
SOURce:VOLTage:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:VOLTage[:STATe]
value: bool = driver.applications.k30NoiseFigure.source.voltage.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: SOURce:VOLTage[:STATe]
driver.applications.k30NoiseFigure.source.voltage.state.set(state = False)
```

No command help available

param state
No help available

6.1.2.10.6.30 UsePort

SCPI Commands

SOURce:VOLTage:USEPort

class UsePortCls

UsePort commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PowerSource

```
# SCPI: SOURce:VOLTage:USEPort
value: enums.PowerSource = driver.applications.k30NoiseFigure.source.voltage.
↪usePort.get()
```

No command help available

return
power_source: No help available

set(power_source: PowerSource) → None

```
# SCPI: SOURce:VOLTage:USEPort
driver.applications.k30NoiseFigure.source.voltage.usePort.set(power_source =
↪enums.PowerSource.VSUPply)
```

No command help available

param power_source
No help available

6.1.2.11 System

class SystemCls

System commands group definition. 11 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.clone()
```

Subgroups

6.1.2.11.1 Communicate

class CommunicateCls

Communicate commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.clone()
```

Subgroups

6.1.2.11.1.1 Gpib

class GpibCls

Gpib commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.gpib.clone()
```

Subgroups

6.1.2.11.1.2 Rdevice

class RdeviceCls

Rdevice commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.gpib.rdevice.clone()
```

Subgroups

6.1.2.11.1.3 Generator

class GeneratorCls

Generator commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.gpib.rdevice.generator.
↳clone()
```

Subgroups

6.1.2.11.1.4 Address

SCPI Commands

```
SYSTem:COMMunicate:GPIB:RDEvice:GENerator:ADDRess
```

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: SYSTem:COMMunicate:GPIB:RDEvice:GENerator:ADDRess
value: int = driver.applications.k30NoiseFigure.system.communicate.gpib.rdevice.
↳generator.address.get()
```

Changes the IEC/IEEE-bus address of the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

return
number: Range: 0 to 30

set(number: int) → None

```
# SCPI: SYSTem:COMMunicate:GPIB:RDEvice:GENerator:ADDRess
driver.applications.k30NoiseFigure.system.communicate.gpib.rdevice.generator.
↳address.set(number = 1)
```

Changes the IEC/IEEE-bus address of the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

param number
Range: 0 to 30

6.1.2.11.1.5 Rdevice

class RdeviceCls

Rdevice commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.rdevice.clone()
```

Subgroups

6.1.2.11.1.6 Generator

class GeneratorCls

Generator commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.rdevice.generator.clone()
```

Subgroups

6.1.2.11.1.7 Interface

SCPI Commands

```
SYSTEM:COMMunicate:RDEVICE:GENerator:INTERface
```

class InterfaceCls

Interface commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → GeneratorIntf

```
# SCPI: SYSTEM:COMMunicate:RDEVICE:GENerator:INTERface
value: enums.GeneratorIntf = driver.applications.k30NoiseFigure.system.
    ↪ communicate.rdevice.generator.interface.get()
```

Defines the interface used for the connection to the external generator.

return
type_py: GPIB TCPip

set(*type_py*: *GeneratorIntf*) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator:INTERface
driver.applications.k30NoiseFigure.system.communicate.rdevice.generator.
↪ interface.set(type_py = enums.GeneratorIntf.GPIB)
```

Defines the interface used for the connection to the external generator.

param *type_py*
GPIB TCPip

6.1.2.11.1.8 Link

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:GENerator:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → GeneratorLink

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator:LINK
value: enums.GeneratorLink = driver.applications.k30NoiseFigure.system.
↪ communicate.rdevice.generator.link.get()
```

No command help available

return
mode: No help available

set(*mode*: *GeneratorLink*) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator:LINK
driver.applications.k30NoiseFigure.system.communicate.rdevice.generator.link.
↪ set(mode = enums.GeneratorLink.GPIB)
```

No command help available

param *mode*
No help available

6.1.2.11.1.9 TypePy

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:GENerator:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator:TYPE
value: str = driver.applications.k30NoiseFigure.system.communicate.rdevice.
↳ generator.typePy.get()
```

This command selects the type of external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

return
name: Generator name as string value

set(name: str) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator:TYPE
driver.applications.k30NoiseFigure.system.communicate.rdevice.generator.typePy.
↳ set(name = '1')
```

This command selects the type of external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

param name
Generator name as string value

6.1.2.11.1.10 TcpiP

class TcpiCls

TcpiP commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.tcpiP.clone()
```

Subgroups

6.1.2.11.1.11 Rdevice

class RdeviceCls

Rdevice commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.tcpiP.rdevice.clone()
```

Subgroups

6.1.2.11.1.12 Generator

class GeneratorCls

Generator commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.communicate.tcpip.rdevice.generator.
↳ clone()
```

Subgroups

6.1.2.11.1.13 Address

SCPI Commands

```
SYSTem:COMMunicate:TCPip:RDEvice:GENerator:ADDRess
```

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:TCPip:RDEvice:GENerator:ADDRess
value: str = driver.applications.k30NoiseFigure.system.communicate.tcpip.
↳ rdevice.generator.address.get()
```

Configures the TCP/IP address for the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

return

address: TCP/IP address between 0.0.0.0 and 0.255.255.255

set(address: str) → None

```
# SCPI: SYSTem:COMMunicate:TCPip:RDEvice:GENerator:ADDRess
driver.applications.k30NoiseFigure.system.communicate.tcpip.rdevice.generator.
↳ address.set(address = '1')
```

Configures the TCP/IP address for the external generator. This command is only valid if External Generator Control (R&S FSWP-B10) is installed.

param address

TCP/IP address between 0.0.0.0 and 0.255.255.255

6.1.2.11.2 Configure

class ConfigureCls

Configure commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.clone()
```

Subgroups

6.1.2.11.2.1 Dut

class DutCls

Dut commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.dut.clone()
```

Subgroups

6.1.2.11.2.2 Gain

SCPI Commands

```
SYSTem:CONFigure:DUT:GAIN
```

class GainCls

Gain commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SYSTem:CONFigure:DUT:GAIN
value: float = driver.applications.k30NoiseFigure.system.configure.dut.gain.
↳get()
```

This command defines the expected ‘gain’ of the DUT. The application uses the ‘gain’ for automatic reference level detection.

return

gain: Range: 10 to 1000, Unit: DB

set(gain: float) → None

```
# SCPI: SYSTem:CONFigure:DUT:GAIN
driver.applications.k30NoiseFigure.system.configure.dut.gain.set(gain = 1.0)
```

This command defines the expected ‘gain’ of the DUT. The application uses the ‘gain’ for automatic reference level detection.

param gain

Range: 10 to 1000, Unit: DB

6.1.2.11.2.3 Stime

SCPI Commands

```
SYSTem:CONFigure:DUT:STIME
```

class StimeCls

Stime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SYSTem:CONFigure:DUT:STIME
value: float = driver.applications.k30NoiseFigure.system.configure.dut.stime.
↳get()
```

This command defines the settling time of the noise source.

return

settling_time: Range: 0 s to 20 s, Unit: S

set(settling_time: float) → None

```
# SCPI: SYSTem:CONFigure:DUT:STIME
driver.applications.k30NoiseFigure.system.configure.dut.stime.set(settling_time,
↳ 1.0)
```

This command defines the settling time of the noise source.

param settling_time

Range: 0 s to 20 s, Unit: S

6.1.2.11.2.4 Generator

class GeneratorCls

Generator commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.generator.clone()
```

Subgroups

6.1.2.11.2.5 Control

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.generator.control.clone()
```

Subgroups

6.1.2.11.2.6 State

SCPI Commands

```
SYSTem:CONFigure:GENerator:CONTRol:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:CONFigure:GENerator:CONTRol:STATe
value: bool = driver.applications.k30NoiseFigure.system.configure.generator.
↳ control.state.get()
```

This command turns automatic control of an external generator on and off. The command is available with option R&S FSWP-B10.

return
state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SYSTem:CONFigure:GENerator:CONTRol:STATe
driver.applications.k30NoiseFigure.system.configure.generator.control.state.
↳ set(state = False)
```

This command turns automatic control of an external generator on and off. The command is available with option R&S FSWP-B10.

param state
ON | OFF | 1 | 0

6.1.2.11.2.7 Initialise

class InitialiseCls

Initialise commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.generator.initialise.clone()
```

Subgroups

6.1.2.11.2.8 Auto

SCPI Commands

```
SYSTem:CONFigure:GENerator:INITialise:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:CONFigure:GENerator:INITialise:AUTO
value: bool = driver.applications.k30NoiseFigure.system.configure.generator.
↳ initialise.auto.get()
```

This command turns automatic connection to the generator on and off. If on, the application automatically configures the generator before each measurement and turns on its RF output. Note that you have to establish a connection to the generator before you can perform the measurement. The command is available with option R&S FSWP-B10.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SYSTem:CONFigure:GENerator:INITialise:AUTO
driver.applications.k30NoiseFigure.system.configure.generator.initialise.auto.
↳ set(state = False)
```

This command turns automatic connection to the generator on and off. If on, the application automatically configures the generator before each measurement and turns on its RF output. Note that you have to establish a connection to the generator before you can perform the measurement. The command is available with option R&S FSWP-B10.

param state

ON | OFF | 1 | 0

6.1.2.11.2.9 Immediate

SCPI Commands

`SYSTem:CONFigure:GENerator:INITialise:IMMediate`

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`set()` → None

```
# SCPI: SYSTem:CONFigure:GENerator:INITialise[:IMMediate]
driver.applications.k30NoiseFigure.system.configure.generator.initialise.
↳ immediate.set()
```

This command establishes a connection to the external generator. When you send the command, the application configures the generator once and turns on its RF output. Note that you have to establish a connection to the generator before you can perform the measurement. The command is available with option R&S FSWP-B10.

`set_with_opc(opc_timeout_ms: int = -1)` → None

```
# SCPI: SYSTem:CONFigure:GENerator:INITialise[:IMMediate]
driver.applications.k30NoiseFigure.system.configure.generator.initialise.
↳ immediate.set_with_opc()
```

This command establishes a connection to the external generator. When you send the command, the application configures the generator once and turns on its RF output. Note that you have to establish a connection to the generator before you can perform the measurement. The command is available with option R&S FSWP-B10.

Same as set, but waits for the operation to complete before continuing further. Use the `RsFswp.utilities.opc_timeout_set()` to set the timeout value.

param `opc_timeout_ms`

Maximum time to wait in milliseconds, valid only for this call.

6.1.2.11.2.10 Switch

class SwitchCls

Switch commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.system.configure.generator.switch.clone()
```

Subgroups

6.1.2.11.2.11 Auto

SCPI Commands

```
SYSTem:CONFigure:GENerator:SWITch:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:CONFigure:GENerator:SWITch:AUTO
value: bool = driver.applications.k30NoiseFigure.system.configure.generator.
↳switch.auto.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:CONFigure:GENerator:SWITch:AUTO
driver.applications.k30NoiseFigure.system.configure.generator.switch.auto.
↳set(state = False)
```

No command help available

param state

No help available

6.1.2.12 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k30NoiseFigure.trace.repcap_window_get()
driver.applications.k30NoiseFigure.trace.repcap_window_set(repcap.Window.Nr1)
```

SCPI Commands

```
TRACe<Window>:COPY
```

class TraceCls

Trace commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Window, default value after init: Window.Nr1

copy(target_trace: TraceTypeK30, source_trace: TraceTypeK30, window=Window.Default) → None

```
# SCPI: TRACe<n>:COPY
driver.applications.k30NoiseFigure.trace.copy(target_trace = enums.TraceTypeK30.
↳TRACe1, source_trace = enums.TraceTypeK30.TRACe1, window = repcap.Window.
↳Default)
```

This command copies data from one trace to another.

param target_trace

No help available

param source_trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trace.clone()
```

Subgroups

6.1.2.12.1 Data

SCPI Commands

```
FORMAT REAL, 32;TRACe<Window>:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_type: TraceTypeK30, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.applications.k30NoiseFigure.trace.data.get(trace_
↳type = enums.TraceTypeK30.TRACe1, window = repcap.Window.Default)
```

This command queries the 'noise figure' measurement results.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_data: For any graphical result display, the command returns one result for each measurement point. The unit depends on the result you are querying.

6.1.2.13 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k30NoiseFigure.trigger.repcap_triggerPort_get()
driver.applications.k30NoiseFigure.trigger.repcap_triggerPort_set(repcap.TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 1 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trigger.clone()
```

Subgroups

6.1.2.13.1 Sequence

class SequenceCls

Sequence commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trigger.sequence.clone()
```

Subgroups

6.1.2.13.1.1 Dtime

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:DTIME
```

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:DTIME
value: float = driver.applications.k30NoiseFigure.trigger.sequence.dtime.
↳get(triggerPort = repcap.TriggerPort.Default)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

dropout_time: Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

set(dropout_time: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:DTIME
driver.applications.k30NoiseFigure.trigger.sequence.dtime.set(dropout_time = 1.
↪0, triggerPort = repcap.TriggerPort.Default)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param dropout_time

Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.2.13.1.2 Holdoff

class HoldoffCls

Holdoff commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trigger.sequence.holdoff.clone()
```

Subgroups

6.1.2.13.1.3 Time

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:HOLDoff:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:HOLDoff[:TIME]
value: float = driver.applications.k30NoiseFigure.trigger.sequence.holdoff.time.
↪get(triggerPort = repcap.TriggerPort.Default)
```

Defines the time offset between the trigger event and the start of the measurement.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

offset: Unit: S

set(offset: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:HOLDoff[:TIME]
driver.applications.k30NoiseFigure.trigger.sequence.holdoff.time.set(offset = 1.
↪0, triggerPort = repcap.TriggerPort.Default)
```

Defines the time offset between the trigger event and the start of the measurement.

param offset

Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.2.13.1.4 Level**class LevelCls**

Level commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trigger.sequence.level.clone()
```

Subgroups**6.1.2.13.1.5 External<ExternalPort>****RepCap Settings**

```
# Range: Nr1 .. Nr3
rc = driver.applications.k30NoiseFigure.trigger.sequence.level.external.repcap_
↪externalPort_get()
driver.applications.k30NoiseFigure.trigger.sequence.level.external.repcap_externalPort_
↪set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:LEVel:EXTeRnal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(triggerPort=TriggerPort.Default, externalPort=ExternalPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel[:EXTeRnal<1/2/3>]
value: float = driver.applications.k30NoiseFigure.trigger.sequence.level.
↳ external.get(triggerPort = repcap.TriggerPort.Default, externalPort = repcap.
↳ ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

trigger_level: Range: 0.5 V to 3.5 V, Unit: V

set(trigger_level: float, triggerPort=TriggerPort.Default, externalPort=ExternalPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel[:EXTeRnal<1/2/3>]
driver.applications.k30NoiseFigure.trigger.sequence.level.external.set(trigger_
↳ level = 1.0, triggerPort = repcap.TriggerPort.Default, externalPort = repcap.
↳ ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param trigger_level

Range: 0.5 V to 3.5 V, Unit: V

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k30NoiseFigure.trigger.sequence.level.external.clone()
```

6.1.2.13.1.6 Slope

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:SLOPe
```

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → SlopeType

```
# SCPI: TRIGger<tp>[:SEquence]:SLOPe
value: enums.SlopeType = driver.applications.k30NoiseFigure.trigger.sequence.
↳slope.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger slope.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

type_py: POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

set(type_py: SlopeType, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:SLOPe
driver.applications.k30NoiseFigure.trigger.sequence.slope.set(type_py = enums.
↳SlopeType.NEGative, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger slope.

param type_py

POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.2.13.1.7 Source

SCPI Commands

`TRIGger<TriggerPort>:SEquence:SOURce`

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*triggerPort*=*TriggerPort.Default*) → ReferenceSourceD

```
# SCPI: TRIGger<tp>[:SEquence]:SOURce
value: enums.ReferenceSourceD = driver.applications.k30NoiseFigure.trigger.
↳sequence.source.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

source: IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’.

set(*source*: *ReferenceSourceD*, *triggerPort*=*TriggerPort.Default*) → None

```
# SCPI: TRIGger<tp>[:SEquence]:SOURce
driver.applications.k30NoiseFigure.trigger.sequence.source.set(source = enums.
↳ReferenceSourceD.EXT2, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

param source

IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.3 K40_PhaseNoise

class K40_PhaseNoiseCls

K40_PhaseNoise commands group definition. 37 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40_PhaseNoise.clone()
```

Subgroups

6.1.3.1 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.calculate.repcap_window_get()
driver.applications.k40PhaseNoise.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 9 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.clone()
```

Subgroups

6.1.3.1.1 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k40PhaseNoise.calculate.deltaMarker.repcap_deltaMarker_get()
driver.applications.k40PhaseNoise.calculate.deltaMarker.repcap_deltaMarker_set(repcap.
↳DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.deltaMarker.clone()
```

Subgroups

6.1.3.1.1.1 Y

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:Y
value: float = driver.applications.k40PhaseNoise.calculate.deltaMarker.y.
get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

level: No help available

6.1.3.1.2 Limit<LimitIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k40PhaseNoise.calculate.limit.repcap_limitIx_get()
driver.applications.k40PhaseNoise.calculate.limit.repcap_limitIx_set(repcap.LimitIx.Nr1)
```

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: LimitIx, default value after init: LimitIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.limit.clone()
```

Subgroups

6.1.3.1.2.1 Active

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[str]

```
# SCPI: CALCulate<n>:LIMit<li>:ACTive
value: List[str] = driver.applications.k40PhaseNoise.calculate.limit.active.
    get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command queries the names of all active limit lines.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

active_limits: No help available

6.1.3.1.3 Marker<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.calculate.marker.repcap_marker_get()
driver.applications.k40PhaseNoise.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.marker.clone()
```

Subgroups

6.1.3.1.3.1 Y

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.applications.k40PhaseNoise.calculate.marker.y.get(window_
↪ = repcap.Window.Default, marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

level: No help available

6.1.3.1.4 PnLimit

class PnLimitCls

PnLimit commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.pnLimit.clone()
```

Subgroups

6.1.3.1.4.1 Auto

SCPI Commands

```
CALCulate<Window>:PNLimit:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:PNLimit:AUTO
driver.applications.k40PhaseNoise.calculate.pnLimit.auto.set(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.3.1.4.2 Fail

SCPI Commands

```
CALCulate<Window>:PNLimit:FAIL
```

class FailCls

Fail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:PNLimit:FAIL
value: bool = driver.applications.k40PhaseNoise.calculate.pnLimit.fail.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_check: No help available

6.1.3.1.4.3 Fc<CornerFrequency>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.applications.k40PhaseNoise.calculate.pnLimit.fc.repcap_cornerFrequency_get()
driver.applications.k40PhaseNoise.calculate.pnLimit.fc.repcap_cornerFrequency_set(repcap.
↳ CornerFrequency.Nr1)
```

SCPI Commands

```
CALCulate<Window>:PNLimit:FC<CornerFrequency>
```

class FcCls

Fc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CornerFrequency, default value after init: CornerFrequency.Nr1

get(window=Window.Default, cornerFrequency=CornerFrequency.Default) → float

```
# SCPI: CALCulate<n>:PNLimit:FC<res>
value: float = driver.applications.k40PhaseNoise.calculate.pnLimit.fc.
↳ get(window = repcap.Window.Default, cornerFrequency = repcap.CornerFrequency.
↳ Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param cornerFrequency

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fc')

return

corner_frequency_value: No help available

set(corner_frequency_value: float, window=Window.Default, cornerFrequency=CornerFrequency.Default) → None

```
# SCPI: CALCulate<n>:PNLimit:FC<res>
driver.applications.k40PhaseNoise.calculate.pnLimit.fc.set(corner_frequency_
↳ value = 1.0, window = repcap.Window.Default, cornerFrequency = repcap.
↳ CornerFrequency.Default)
```

No command help available

param corner_frequency_value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param cornerFrequency

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fc')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.pnLimit.fc.clone()
```

6.1.3.1.5 Snoise<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.calculate.snoise.repcap_marker_get()
driver.applications.k40PhaseNoise.calculate.snoise.repcap_marker_set(repcap.Marker.Nr1)
```

class SnoiseCls

Snoise commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.snoise.clone()
```

Subgroups

6.1.3.1.5.1 Decades

class DecadesCls

Decades commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.calculate.snoise.decades.clone()
```

Subgroups

6.1.3.1.5.2 X

SCPI Commands

```
CALCulate<Window>:SNOise<Marker>:DECades:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → List[float]

```
# SCPI: CALCulate<n>:SNOise<m>:DECades:X
value: List[float] = driver.applications.k40PhaseNoise.calculate.snoise.decades.
↳ x.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Snoise’)

return

frequencies: No help available

6.1.3.1.5.3 Y

SCPI Commands

```
CALCulate<Window>:SNOise<Marker>:DECades:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → List[float]

```
# SCPI: CALCulate<n>:SNOise<m>:DECades:Y
value: List[float] = driver.applications.k40PhaseNoise.calculate.snoise.decades.
↳ y.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Snoise’)

return

levels: No help available

6.1.3.1.5.4 Y

SCPI Commands

```
CALCulate<Window>:SNOise<Marker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:SNOise<m>:Y
value: float = driver.applications.k40PhaseNoise.calculate.snoise.y.get(window_
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Snoise')

return

level: No help available

6.1.3.2 Display

class DisplayCls

Display commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.display.clone()
```

Subgroups

6.1.3.2.1 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.display.window.repcap_window_get()
driver.applications.k40PhaseNoise.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.display.window.clone()
```

Subgroups

6.1.3.2.1.1 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.applications.k40PhaseNoise.display.window.trace.repcap_trace_get()
driver.applications.k40PhaseNoise.display.window.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability:
Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.display.window.trace.clone()
```

Subgroups

6.1.3.2.1.2 Select

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(trace_number: int, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SElect
driver.applications.k40PhaseNoise.display.window.trace.select.set(trace_number,
↪= 1, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param trace_number

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface
'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.3.3 Fetch**class FetchCls**

Fetch commands group definition. 20 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.clone()
```

Subgroups**6.1.3.3.1 Pnoise<Trace>****RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.applications.k40PhaseNoise.fetch.pnoise.repcap_trace_get()
driver.applications.k40PhaseNoise.fetch.pnoise.repcap_trace_set(repcap.Trace.Tr1)
```

class PnoiseCls

Pnoise commands group definition. 20 total commands, 8 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.pnoise.clone()
```

Subgroups**6.1.3.3.1.1 Ipn****SCPI Commands**

```
FETCh:PNOise<Trace>:IPN
```

class IpnCls

Ipn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → float

```
# SCPI: FETCh:PNOise<t>:IPN
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.ipn.get(trace = ↵
↵repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

value: No help available

6.1.3.3.1.2 Measured

class MeasuredCls

Measured commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.pnoise.measured.clone()
```

Subgroups

6.1.3.3.1.3 Frequency

SCPI Commands

```
FETCh:PNOise<Trace>:MEASured:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → float

```
# SCPI: FETCh:PNOise<t>:MEASured:FREQuency
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.measured.
↵frequency.get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

frequency: No help available

6.1.3.3.1.4 Level

SCPI Commands

```
FETCH:PNOise<Trace>:MEASured:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCH:PNOise<t>:MEASured:LEVel
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.measured.level.
↪get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

level: No help available

6.1.3.3.1.5 Rfm

SCPI Commands

```
FETCH:PNOise<Trace>:RFM
```

class RfmCls

Rfm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCH:PNOise<t>:RFM
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.rfm.get(trace = ↪
↪repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

value: No help available

6.1.3.3.1.6 Rms

SCPI Commands

FETCh:PNOise<Trace>:RMS

class RmsCls

Rms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCh:PNOise<t>:RMS
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.rms.get(trace = ↵
↵repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

value: No help available

6.1.3.3.1.7 Rpm

SCPI Commands

FETCh:PNOise<Trace>:RPM

class RpmCls

Rpm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCh:PNOise<t>:RPM
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.rpm.get(trace = ↵
↵repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

value: No help available

6.1.3.3.1.8 Spurs

SCPI Commands

```
FETCH:PNOise<Trace>:SPURs
```

class SpursCls

Spurs commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(*trace=Trace.Default*) → List[float]

```
# SCPI: FETCH:PNOise<t>:SPURs
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.spurs.
    ↪get(trace = repcap.Trace.Default)
```

This command queries the location and level of all spurs that have been detected.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

spurs: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.pnoise.spurs.clone()
```

Subgroups

6.1.3.3.1.9 Discrete

SCPI Commands

```
FETCH:PNOise<Trace>:SPURs:DIScrete
```

class DiscreteCls

Discrete commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCH:PNOise<t>:SPURs:DIScrete
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.spurs.discrete.
    ↪get(trace = repcap.Trace.Default)
```

This command queries the discrete jitter result.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return
jitter: numeric value Unit: s

6.1.3.3.1.10 Random

SCPI Commands

`FETCH:PNOise<Trace>:SPURs:RANDOM`

class RandomCls

Random commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: FETCH:PNOise<t>:SPURs:RANDOM
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.spurs.random.
↳ get(trace = repcap.Trace.Default)
```

This command queries the random jitter result.

param trace
optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Pnoise’)

return
jitter: No help available

6.1.3.3.1.11 Sweep

class SweepCls

Sweep commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.clone()
```

Subgroups

6.1.3.3.1.12 Avg

SCPI Commands

`FETCH:PNOise<Trace>:SWEep:AVG`

class AvgCls

Avg commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace*=*Trace.Default*) → List[int]

```
# SCPI: FETCH:PNOise<t>:SWEep:AVG
value: List[int] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.avg.
↪ get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

measurements: No help available

6.1.3.3.1.13 Fdrift

SCPI Commands

```
FETCH:PNOise<Trace>:SWEep:FDRift
```

class FdriftCls

Fdrift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace*=*Trace.Default*) → List[float]

```
# SCPI: FETCH:PNOise<t>:SWEep:FDRift
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.
↪ fdrift.get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

frequency_drifts: No help available

6.1.3.3.1.14 Ldrift

SCPI Commands

```
FETCH:PNOise<Trace>:SWEep:LDRift
```

class LdriftCls

Ldrift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace*=*Trace.Default*) → List[float]

```
# SCPI: FETCH:PNOise<t>:SWEep:LDRift
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.
↪ ldrift.get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

level_drifts: No help available

6.1.3.3.1.15 Mdrift

SCPI Commands

FETCh:PNOise<Trace>:SWEep:MDRift

class MdriftCls

Mdrift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → List[float]

```
# SCPI: FETCh:PNOise<t>:SWEep:MDRift
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.
↳mdrift.get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return

frequency_drifts: No help available

6.1.3.3.1.16 Start

SCPI Commands

FETCh:PNOise<Trace>:SWEep:START

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → List[float]

```
# SCPI: FETCh:PNOise<t>:SWEep:START
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.start.
↳get(trace = repcap.Trace.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return
start_freq_offsets: No help available

6.1.3.3.1.17 Stop

SCPI Commands

```
FETCH:PNOise<Trace>:SWEep:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → List[float]

```
# SCPI: FETCH:PNOise<t>:SWEep:STOP
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.stop.
↳get(trace = repcap.Trace.Default)
```

No command help available

param trace
optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return
freq_offsets: No help available

6.1.3.3.1.18 SymbolRate

SCPI Commands

```
FETCH:PNOise<Trace>:SWEep:SRATe
```

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → List[float]

```
# SCPI: FETCH:PNOise<t>:SWEep:SRATe
value: List[float] = driver.applications.k40PhaseNoise.fetch.pnoise.sweep.
↳symbolRate.get(trace = repcap.Trace.Default)
```

No command help available

param trace
optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

return
sampling_rates: No help available

6.1.3.3.1.19 User<UserRange>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.fetch.pnoise.user.repcap_userRange_get()
driver.applications.k40PhaseNoise.fetch.pnoise.user.repcap_userRange_set(repcap.
↳ UserRange.Nr1)
```

class UserCls

User commands group definition. 4 total commands, 4 Subgroups, 0 group commands Repeated Capability: UserRange, default value after init: UserRange.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.fetch.pnoise.user.clone()
```

Subgroups

6.1.3.3.1.20 Ipn

SCPI Commands

```
FETCH:PNOise<Trace>:USER<UserRange>:IPN
```

class IpnCls

Ipn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default, userRange=UserRange.Default) → float

```
# SCPI: FETCH:PNOise<t>:USER<range>:IPN
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.user.ipn.
↳ get(trace = repcap.Trace.Default, userRange = repcap.UserRange.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

param userRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return

value: No help available

6.1.3.3.1.21 Rfm

SCPI Commands

```
FETCH:PNOise<Trace>:USER<UserRange>:RFM
```

class RfmCls

Rfm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default, userRange=UserRange.Default*) → float

```
# SCPI: FETCH:PNOise<t>:USER<range>:RFM
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.user.rfm.
↪get(trace = repcap.Trace.Default, userRange = repcap.UserRange.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

param userRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return

value: No help available

6.1.3.3.1.22 Rms

SCPI Commands

```
FETCH:PNOise<Trace>:USER<UserRange>:RMS
```

class RmsCls

Rms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default, userRange=UserRange.Default*) → float

```
# SCPI: FETCH:PNOise<t>:USER<range>:RMS
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.user.rms.
↪get(trace = repcap.Trace.Default, userRange = repcap.UserRange.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

param userRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return

value: No help available

6.1.3.3.1.23 Rpm

SCPI Commands

`FETCH:PNOise<Trace>:USER<UserRange>:RPM`

class RpmCls

Rpm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default, userRange=UserRange.Default*) → float

```
# SCPI: FETCH:PNOise<t>:USER<range>:RPM
value: float = driver.applications.k40PhaseNoise.fetch.pnoise.user.rpm.
↪get(trace = repcap.Trace.Default, userRange = repcap.UserRange.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Pnoise')

param userRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'User')

return

value: No help available

6.1.3.4 Layout

class LayoutCls

Layout commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.layout.clone()
```

Subgroups

6.1.3.4.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.layout.add.clone()
```

Subgroups

6.1.3.4.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK40) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k40PhaseNoise.layout.add.window.get(window_
↳ name = '1', direction = enums.WindowDirection.ABOVe, window_type = enums.
↳ WindowTypeK40.FrequencyDrift=FDRift)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method RsFswp.Layout.Replace.Window.set command.

param window_name

String containing the name of the existing window the new window is inserted next to.

By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_query**.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.3.4.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.layout.catalog.clone()
```

Subgroups

6.1.3.4.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k40PhaseNoise.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return

result: No help available

6.1.3.4.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.layout.identify.clone()
```

Subgroups

6.1.3.4.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k40PhaseNoise.layout.identify.window.
    get(window_name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name

String containing the name of a window.

return

window_index: Index number of the window.

6.1.3.4.4 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.layout.replace.clone()
```

Subgroups

6.1.3.4.4.1 Window

SCPI Commands

```
LAYout:REPLace:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeK40) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k40PhaseNoise.layout.replace.window.set(window_name = '1',
↳ window_type = enums.WindowTypeK40.FrequencyDrift=FDRift)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method RsFswp.Layout. **Add.Window.get_** command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method **RsFswp.Layout.Catalog.Window.get_** query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method **RsFswp.Layout.Add.Window.get_** for a list of available window types.

6.1.3.5 Sense

class SenseCls

Sense commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.sense.clone()
```

Subgroups

6.1.3.5.1 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k40PhaseNoise.sense.probe.repcap_probe_get()
driver.applications.k40PhaseNoise.sense.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.sense.probe.clone()
```

Subgroups

6.1.3.5.1.1 Id

class IdCls

Id commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.sense.probe.id.clone()
```

Subgroups

6.1.3.5.1.2 SrNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:SRNumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → str

```
# SCPI: [SENSe]:PROBe<pb>:ID:SRNumber
value: str = driver.applications.k40PhaseNoise.sense.probe.id.srNumber.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

serial_no: No help available

6.1.3.6 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k40PhaseNoise.trace.repcap_window_get()
driver.applications.k40PhaseNoise.trace.repcap_window_set(repcap.Window.Nr1)
```

SCPI Commands

```
TRACe<Window>:COPY
```

class TraceCls

Trace commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Window, default value after init: Window.Nr1

copy(*target_trace: TraceTypeNumeric, source_trace: TraceTypeNumeric, window=Window.Default*) → None

```
# SCPI: TRACe<n>:COPY
driver.applications.k40PhaseNoise.trace.copy(target_trace = enums.
↳TraceTypeNumeric.TRACe1, source_trace = enums.TraceTypeNumeric.TRACe1, window_
↳= repcap.Window.Default)
```

This command copies data from one trace to another.

param target_trace

No help available

param source_trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k40PhaseNoise.trace.clone()
```

Subgroups

6.1.3.6.1 Data

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(trace_type: TraceTypeNumeric, window=Window.Default) → List[float]`

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.applications.k40PhaseNoise.trace.data.get(trace_
↪type = enums.TraceTypeNumeric.TRACe1, window = repcap.Window.Default)
```

This command queries the trace data (measurement results) .

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_data: No help available

6.1.4 K50_Spurious

class K50_SpuriousCls

K50_Spurious commands group definition. 235 total commands, 16 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50_Spurious.clone()
```

Subgroups

6.1.4.1 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calculate.repcap_window_get()
driver.applications.k50Spurious.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 43 total commands, 8 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.clone()
```

Subgroups

6.1.4.1.1 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k50Spurious.calculate.deltaMarker.repcap_deltaMarker_get()
driver.applications.k50Spurious.calculate.deltaMarker.repcap_deltaMarker_set(repcap.
↳DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 16 total commands, 9 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.clone()
```

Subgroups

6.1.4.1.1.1 Aoff

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:AOFF
driver.applications.k50Spurious.calculate.deltaMarker.aoff.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns off all delta markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.4.1.1.2 LinkTo**class LinkToCls**

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.linkTo.clone()
```

Subgroups**6.1.4.1.1.3 Marker<MarkerDestination>****RepCap Settings**

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calculate.deltaMarker.linkTo.marker.recap_
↪markerDestination_get()
driver.applications.k50Spurious.calculate.deltaMarker.linkTo.marker.recap_
↪markerDestination_set(repcap.MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:LINK:TO:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

get(window=Window.Default, deltaMarker=DeltaMarker.Default, markerDestination=MarkerDestination.Default) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m1>:LINK:TO:MARKer<m2>
value: bool = driver.applications.k50Spurious.calculate.deltaMarker.linkTo.
↪marker.get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default, markerDestination = repcap.MarkerDestination.Default)
```

This command links the delta source marker <ms> to any active destination marker <md> (normal or delta marker) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m1>:LINK:TO:MARKer<m2>
driver.applications.k50Spurious.calculate.deltaMarker.linkTo.marker.set(state = False, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.  
Default, markerDestination = repcap.MarkerDestination.Default)
```

This command links the delta source marker <ms> to any active destination marker <md> (normal or delta marker) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.linkTo.marker.clone()
```

6.1.4.1.1.4 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.maximum.clone()
```

Subgroups

6.1.4.1.1.5 Left

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:LEFT
driver.applications.k50Spurious.calculate.deltaMarker.maximum.left.set(window = ↪
↪repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.6 Next

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:NEXT
driver.applications.k50Spurious.calculate.deltaMarker.maximum.next.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next positive peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.7 Peak

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum[:PEAK]
driver.applications.k50Spurious.calculate.deltaMarker.maximum.peak.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.8 Right

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:RIGHT
driver.applications.k50Spurious.calculate.deltaMarker.maximum.right.set(window_
↳ repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value on the trace. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.9 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.minimum.clone()
```

Subgroups

6.1.4.1.1.10 Left

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MINimum:LEFT
driver.applications.k50Spurious.calculate.deltaMarker.minimum.left.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.11 Next

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MINimum:NEXT
driver.applications.k50Spurious.calculate.deltaMarker.minimum.next.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.12 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum[:PEAK]
driver.applications.k50Spurious.calculate.deltaMarker.minimum.peak.set(window = ↵
↵= repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.13 Right

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:RIGHT
driver.applications.k50Spurious.calculate.deltaMarker.minimum.right.set(window ↵
↵= repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.1.14 Mref**SCPI Commands**

CALCulate<Window>:DELTamarker<DeltaMarker>:MREF

class MrefCls

Mref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → int

<pre># SCPI: CALCulate<n>:DELTamarker<m>:MREF value: int = driver.applications.k50Spurious.calculate.deltaMarker.mref. ↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)</pre>

This command selects a reference marker for a delta marker other than marker 1. The reference may be another marker or the fixed reference.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

ref_marker: No help available

set(ref_marker: int, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

<pre># SCPI: CALCulate<n>:DELTamarker<m>:MREF driver.applications.k50Spurious.calculate.deltaMarker.mref.set(ref_marker = 1,↳ ↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)</pre>
--

This command selects a reference marker for a delta marker other than marker 1. The reference may be another marker or the fixed reference.

param ref_marker

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.1.4.1.1.15 State

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
value: bool = driver.applications.k50Spurious.calculate.deltaMarker.state.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
driver.applications.k50Spurious.calculate.deltaMarker.state.set(state = False,
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.4.1.1.16 Trace

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:TRACe`

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
value: float = driver.applications.k50Spurious.calculate.deltaMarker.trace.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

trace: Trace number the marker is assigned to.

set(trace: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
driver.applications.k50Spurious.calculate.deltaMarker.trace.set(trace = 1.0,
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

Trace number the marker is assigned to.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.4.1.1.17 X

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:X

class XCls

X commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
value: float = driver.applications.k50Spurious.calculate.deltaMarker.x.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

stimulus: No help available

set(stimulus: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
driver.applications.k50Spurious.calculate.deltaMarker.x.set(stimulus = 1.0,↳
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.deltaMarker.x.clone()
```

Subgroups

6.1.4.1.1.18 Relative

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:X:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X:RELative
value: float = driver.applications.k50Spurious.calculate.deltaMarker.x.relative.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command queries the relative position of a delta marker on the x-axis. If necessary, the command activates the delta marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

xvalue_relative: No help available

6.1.4.1.1.19 Y

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:Y
value: float = driver.applications.k50Spurious.calculate.deltaMarker.y.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

value: No help available

6.1.4.1.2 Dline<DisplayLine>**RepCap Settings**

```
# Range: Nr1 .. Nr2
rc = driver.applications.k50Spurious.calculate.dline.repcap_displayLine_get()
driver.applications.k50Spurious.calculate.dline.repcap_displayLine_set(repcap.
↪DisplayLine.Nr1)
```

SCPI Commands

```
CALCulate<Window>:DLINe<DisplayLine>
```

class DlineCls

Dline commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: DisplayLine, default value after init: DisplayLine.Nr1

get(window=Window.Default, displayLine=DisplayLine.Default) → float

```
# SCPI: CALCulate<n>:DLINe<dl>
value: float = driver.applications.k50Spurious.calculate.dline.get(window = ↪
↪repcap.Window.Default, displayLine = repcap.DisplayLine.Default)
```

This command defines the (horizontal) position of a display line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param displayLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dline')

return

line: No help available

set(line: float, window=Window.Default, displayLine=DisplayLine.Default) → None

```
# SCPI: CALCulate<n>:DLINe<dl>
driver.applications.k50Spurious.calculate.dline.set(line = 1.0, window = repcap.
↪Window.Default, displayLine = repcap.DisplayLine.Default)
```

This command defines the (horizontal) position of a display line.

param line

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param displayLine

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Dline’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.dline.clone()
```

Subgroups

6.1.4.1.2.1 State

SCPI Commands

```
CALCulate<Window>:DLINe<DisplayLine>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, displayLine=DisplayLine.Default) → bool

```
# SCPI: CALCulate<n>:DLINe<dl>:STATe
value: bool = driver.applications.k50Spurious.calculate.dline.state.get(window_
↳= repcap.Window.Default, displayLine = repcap.DisplayLine.Default)
```

This command turns a display line on and off

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param displayLine

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Dline’)

return

arg_0: No help available

set(arg_0: bool, window=Window.Default, displayLine=DisplayLine.Default) → None

```
# SCPI: CALCulate<n>:DLINe<dl>:STATe
driver.applications.k50Spurious.calculate.dline.state.set(arg_0 = False, window_
↳= repcap.Window.Default, displayLine = repcap.DisplayLine.Default)
```

This command turns a display line on and off

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param displayLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dline')

6.1.4.1.3 Fline<FreqLine>**RepCap Settings**

```
# Range: Nr1 .. Nr4
rc = driver.applications.k50Spurious.calculate.fline.repcap_freqLine_get()
driver.applications.k50Spurious.calculate.fline.repcap_freqLine_set(repcap.FreqLine.Nr1)
```

SCPI Commands

```
CALCulate<Window>:FLINE<FreqLine>
```

class FlineCls

Fline commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: FreqLine, default value after init: FreqLine.Nr1

get(window=Window.Default, freqLine=FreqLine.Default) → float

```
# SCPI: CALCulate<n>:FLINE<dl>
value: float = driver.applications.k50Spurious.calculate.fline.get(window = ↵
↵repcap.Window.Default, freqLine = repcap.FreqLine.Default)
```

This command defines the position of a frequency line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param freqLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fline')

return

arg_0: No help available

set(arg_0: float, window=Window.Default, freqLine=FreqLine.Default) → None

```
# SCPI: CALCulate<n>:FLINE<dl>
driver.applications.k50Spurious.calculate.fline.set(arg_0 = 1.0, window = ↵
↵repcap.Window.Default, freqLine = repcap.FreqLine.Default)
```

This command defines the position of a frequency line.

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param freqLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fline')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.fline.clone()
```

Subgroups

6.1.4.1.3.1 State

SCPI Commands

```
CALCulate<Window>:FLINe<FreqLine>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, freqLine=FreqLine.Default) → bool

```
# SCPI: CALCulate<n>:FLINe<dl>:STATe
value: bool = driver.applications.k50Spurious.calculate.fline.state.get(window_
↳= repcap.Window.Default, freqLine = repcap.FreqLine.Default)
```

This command turns a frequency line on and off

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param freqLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fline')

return

arg_0: No help available

set(arg_0: bool, window=Window.Default, freqLine=FreqLine.Default) → None

```
# SCPI: CALCulate<n>:FLINe<dl>:STATe
driver.applications.k50Spurious.calculate.fline.state.set(arg_0 = False, window_
↳= repcap.Window.Default, freqLine = repcap.FreqLine.Default)
```

This command turns a frequency line on and off

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param freqLine

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Fline')

6.1.4.1.4 Limit<LimitIx>**RepCap Settings**

```
# Range: Nr1 .. Nr32
rc = driver.applications.k50Spurious.calculate.limit.repcap_limitIx_get()
driver.applications.k50Spurious.calculate.limit.repcap_limitIx_set(repcap.LimitIx.Nr1)
```

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: LimitIx, default value after init: LimitIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.limit.clone()
```

Subgroups**6.1.4.1.4.1 Clear****class ClearCls**

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.limit.clear.clone()
```

Subgroups

6.1.4.1.4.2 Immediate

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:CLEar:IMMediate`

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CLEar[:IMMediate]
driver.applications.k50Spurious.calculate.limit.clear.immediate.set(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command deletes the result of the current limit check. The command works on all limit lines in all measurement windows at the same time.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

set_with_opc(window=Window.Default, limitIx=LimitIx.Default, opc_timeout_ms: int = -1) → None

6.1.4.1.4.3 Fail

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:FAIL`

class FailCls

Fail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:FAIL
value: bool = driver.applications.k50Spurious.calculate.limit.fail.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command queries the result of a limit check in the specified window. To get a valid result, you have to perform a complete measurement with synchronization to the end of the measurement before reading out the result. This is only possible for single measurement mode.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

result: 0 PASS 1 FAIL

6.1.4.1.5 Marker<Marker>**RepCap Settings**

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calculate.marker.repcap_marker_get()
driver.applications.k50Spurious.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 16 total commands, 10 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.marker.clone()
```

Subgroups**6.1.4.1.5.1 Aoff****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:AOFF
driver.applications.k50Spurious.calculate.marker.aoff.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command turns off all markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.4.1.5.2 Link

SCPI Commands

`CALCulate<Window>:MARKer:LINK`

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer:LINK
value: bool = driver.applications.k50Spurious.calculate.marker.link.get(window_
↳= repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=*Window.Default*) → None

```
# SCPI: CALCulate<n>:MARKer:LINK
driver.applications.k50Spurious.calculate.marker.link.set(state = False, window_
↳= repcap.Window.Default)
```

No command help available

param state

1..n Window

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.4.1.5.3 LinkTo

class LinkToCls

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.marker.linkTo.clone()
```

Subgroups

6.1.4.1.5.4 Marker<MarkerDestination>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calculate.marker.linkTo.marker.repcap_
    ↪markerDestination_get()
driver.applications.k50Spurious.calculate.marker.linkTo.marker.repcap_markerDestination_
    ↪set(repcap.MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LINK:T0:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

get(window=Window.Default, marker=Marker.Default, markerDestination=MarkerDestination.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m1>:LINK:T0:MARKer<m2>
value: bool = driver.applications.k50Spurious.calculate.marker.linkTo.marker.
    ↪get(window = repcap.Window.Default, marker = repcap.Marker.Default, ↪
    ↪markerDestination = repcap.MarkerDestination.Default)
```

This command links the normal source marker <ms> to any active destination marker <md> (normal or delta marker) . If you change the horizontal position of marker <md>, marker <ms> changes its horizontal position to the same value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m1>:LINK:T0:MARKer<m2>
driver.applications.k50Spurious.calculate.marker.linkTo.marker.set(state = ↪
```

(continues on next page)

(continued from previous page)

```
↪False, window = repcap.Window.Default, marker = repcap.Marker.Default,↪
↪markerDestination = repcap.MarkerDestination.Default)
```

This command links the normal source marker <ms> to any active destination marker <md> (normal or delta marker) . If you change the horizontal position of marker <md>, marker <ms> changes its horizontal position to the same value.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.marker.linkTo.marker.clone()
```

6.1.4.1.5.5 Maximum**class MaximumCls**

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.marker.maximum.clone()
```

Subgroups**6.1.4.1.5.6 Left****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:LEFT
driver.applications.k50Spurious.calculate.marker.maximum.left.set(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.7 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:NEXT
driver.applications.k50Spurious.calculate.marker.maximum.next.set(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.8 Peak

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum[:PEAK]
driver.applications.k50Spurious.calculate.marker.maximum.peak.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.9 Right

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:RIGHT
driver.applications.k50Spurious.calculate.marker.maximum.right.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.10 Minimum**class MinimumCls**

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.marker.minimum.clone()
```

Subgroups**6.1.4.1.5.11 Left****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:LEFT
driver.applications.k50Spurious.calculate.marker.minimum.left.set(window = _
↪repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.12 Next

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:NEXT
driver.applications.k50Spurious.calculate.marker.minimum.next.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.13 Peak

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum[:PEAK]
driver.applications.k50Spurious.calculate.marker.minimum.peak.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.14 Right

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:RIGHT
driver.applications.k50Spurious.calculate.marker.minimum.right.set(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.1.5.15 Pexcursion

SCPI Commands

```
CALCulate<Window>:MARKer:PEXCursion
```

class PexcursionCls

Pexcursion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:MARKer:PEXCursion
value: float = driver.applications.k50Spurious.calculate.marker.pexcursion.
↳get(window = repcap.Window.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

excursion: No help available

set(*excursion: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:MARKer:PEXCursion
driver.applications.k50Spurious.calculate.marker.pexcursion.set(excursion = 1.0,
↪ window = repcap.Window.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param excursion

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.4.1.5.16 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
value: bool = driver.applications.k50Spurious.calculate.marker.state.get(window,
↪ = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
driver.applications.k50Spurious.calculate.marker.state.set(state = False,
↪ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.4.1.5.17 Trace**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:TRACe
--

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

<pre># SCPI: CALCulate<n>:MARKer<m>:TRACe value: float = driver.applications.k50Spurious.calculate.marker.trace. ↳ get(window = repcap.Window.Default, marker = repcap.Marker.Default)</pre>
--

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

trace: No help available

set(trace: float, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:TRACe driver.applications.k50Spurious.calculate.marker.trace.set(trace = 1.0, window_ ↳ = repcap.Window.Default, marker = repcap.Marker.Default)</pre>
--

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.4.1.5.18 X

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:X`

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X
value: float = driver.applications.k50Spurious.calculate.marker.x.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

stimulus: No help available

set(*stimulus: float, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X
driver.applications.k50Spurious.calculate.marker.x.set(stimulus = 1.0, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.1.4.1.5.19 Y

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.applications.k50Spurious.calculate.marker.y.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

value: No help available

6.1.4.1.6 PeakSearch

class PeakSearchCls

PeakSearch commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.peakSearch.clone()
```

Subgroups

6.1.4.1.6.1 Pshow

SCPI Commands

```
CALCulate<Window>:PSEarch:PSHow
```

class PshowCls

Pshow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:PSEarch:PSHow
value: bool = driver.applications.k50Spurious.calculate.peakSearch.pshow.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:PSEarch:PSHow
driver.applications.k50Spurious.calculate.peakSearch.pshow.set(state = False,
↳window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.4.1.7 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calculate.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.calculate.pmeter.repcap_powerMeter_set(repcap.PowerMeter.
↳Nr1)
```

class PmeterCls

Pmeter commands group definition. 3 total commands, 1 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.pmeter.clone()
```


Subgroups

6.1.4.1.7.1 Relative

class RelativeCls

Relative commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.pmeter.relative.clone()
```

Subgroups

6.1.4.1.7.2 Magnitude

SCPI Commands

```
CALCulate<Window>:PMETer<PowerMeter>:RELative:MAGNitude
```

class MagnitudeCls

Magnitude commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, powerMeter=PowerMeter.Default) → float

```
# SCPI: CALCulate<n>:PMETer<p>:RELative[:MAGNitude]
value: float = driver.applications.k50Spurious.calculate.pmeter.relative.
↳ magnitude.get(window = repcap.Window.Default, powerMeter = repcap.PowerMeter.
↳ Default)
```

This command defines the reference value for relative measurements.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: float, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALCulate<n>:PMETer<p>:RELative[:MAGNitude]
driver.applications.k50Spurious.calculate.pmeter.relative.magnitude.set(arg_0 =
↳ 1.0, window = repcap.Window.Default, powerMeter = repcap.PowerMeter.Default)
```

This command defines the reference value for relative measurements.

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pmeter’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.pmeter.relative.magnitude.clone()
```

Subgroups

6.1.4.1.7.3 Auto

SCPI Commands

```
CALCulate<Window>:PMETer<PowerMeter>:RELative:MAGNitude:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(arg_0: EventOnce, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALCulate<n>:PMETer<p>:RELative[:MAGNitude]:AUTO
driver.applications.k50Spurious.calculate.pmeter.relative.magnitude.auto.
↳ set(arg_0 = enums.EventOnce.ONCE, window = repcap.Window.Default, powerMeter_
↳ = repcap.PowerMeter.Default)
```

This command sets the current measurement result as the reference level for relative measurements.

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pmeter’)

6.1.4.1.7.4 State

SCPI Commands

```
CALCulate<Window>:PMETer<PowerMeter>:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, powerMeter=PowerMeter.Default) → bool

```
# SCPI: CALCulate<n>:PMETer<p>:RELative:STATe
value: bool = driver.applications.k50Spurious.calculate.pmeter.relative.state.
↳get(window = repcap.Window.Default, powerMeter = repcap.PowerMeter.Default)
```

This command turns relative power sensor measurements on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: bool, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALCulate<n>:PMETer<p>:RELative:STATe
driver.applications.k50Spurious.calculate.pmeter.relative.state.set(arg_0 =
↳False, window = repcap.Window.Default, powerMeter = repcap.PowerMeter.Default)
```

This command turns relative power sensor measurements on and off.

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.1.8 Ssearch

class SsearchCls

Ssearch commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.ssearch.clone()
```

Subgroups

6.1.4.1.8.1 Table

class TableCls

Table commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calculate.ssearch.table.clone()
```

Subgroups

6.1.4.1.8.2 Column

SCPI Commands

```
CALCulate:SSEarch:TABLE:COLumn
```

class ColumnCls

Column commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[HeadersK50]

```
# SCPI: CALCulate:SSEarch:TABLE:COLumn
value: List[enums.HeadersK50] = driver.applications.k50Spurious.calculate.
    ↪ ssearch.table.column.get()
```

Select the numerical results to be displayed in the Spurious Detection Table. For a description of the individual results see ‘Spurious Detection Table’.

return

headers: ALL | SID | START | STOP | RBW | FREQuency | POWER | DELTa | IDENT
ALL All available results are displayed START Start frequency of range/span STOP
Stop frequency of range/span FREQuency Spur frequency POWER Spur power DELTa
Delta of spur to limit RBW Resolution bandwidth used for range IDENT Spur ID

set(state: bool, headers: List[HeadersK50]) → None

```
# SCPI: CALCulate:SSEarch:TABLE:COLumn
driver.applications.k50Spurious.calculate.ssearch.table.column.set(state = False, headers = [HeadersK50.ALL, HeadersK50.STOP])
```

Select the numerical results to be displayed in the Spurious Detection Table. For a description of the individual results see ‘Spurious Detection Table’.

param state

ON | OFF | 0 | 1 OFF | 0 Hides the result ON | 1 Displays the result

param headers

ALL | SID | START | STOP | RBW | FREQuency | POWer | DELTa | IDENT ALL
All available results are displayed START Start frequency of range/span STOP Stop
frequency of range/span FREQuency Spur frequency POWer Spur power DELTa Delta
of spur to limit RBW Resolution bandwidth used for range IDENT Spur ID

6.1.4.2 Calibration

class CalibrationCls

Calibration commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calibration.clone()
```

Subgroups

6.1.4.2.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.calibration.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.calibration.pmeter.repcap_powerMeter_set(repcap.
    PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability:
PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calibration.pmeter.clone()
```

Subgroups

6.1.4.2.1.1 Zero

class ZeroCls

Zero commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.calibration.pmeter.zero.clone()
```

Subgroups

6.1.4.2.1.2 Auto

SCPI Commands

```
CALibration:PMETer<PowerMeter>:ZERO:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(arg_0: EventOnce, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALibration:PMETer<p>:ZERO:AUTO
driver.applications.k50Spurious.calibration.pmeter.zero.auto.set(arg_0 = enums.
↳EventOnce.ONCE, powerMeter = repcap.PowerMeter.Default)
```

This command zeroes the power sensor. Note that you have to disconnect the signals from the power sensor input before you start to zero the power sensor. Otherwise, results are invalid.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.3 Display

class DisplayCls

Display commands group definition. 15 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.clone()
```

Subgroups

6.1.4.3.1 Mtable

SCPI Commands

DISPlay:MTABLE

class MtableCls

Mtable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoMode

```
# SCPI: DISPlay:MTABLE
value: enums.AutoMode = driver.applications.k50Spurious.display.mtable.get()
```

No command help available

```
return
    display_mode: No help available
```

set(display_mode: AutoMode) → None

```
# SCPI: DISPlay:MTABLE
driver.applications.k50Spurious.display.mtable.set(display_mode = enums.
    ↪AutoMode.AUTO)
```

No command help available

```
param display_mode
    No help available
```

6.1.4.3.2 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.display.window.repcap_window_get()
driver.applications.k50Spurious.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 13 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.clone()
```

Subgroups**6.1.4.3.2.1 Minfo****class MinfoCls**

Minfo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.minfo.clone()
```

Subgroups**6.1.4.3.2.2 State****SCPI Commands**

```
DISPlay:WINDow<Window>:MINFo:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
value: bool = driver.applications.k50Spurious.display.window.minfo.state.
↳get(window = repcap.Window.Default)
```

This command turns the marker information in all diagrams on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: ON | 1 Displays the marker information in the diagrams. OFF | 0 Hides the marker information in the diagrams.

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
driver.applications.k50Spurious.display.window.minfo.state.set(state = False,
↪window = repcap.Window.Default)
```

This command turns the marker information in all diagrams on and off.

param state

ON | 1 Displays the marker information in the diagrams. OFF | 0 Hides the marker information in the diagrams.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.3 Mtable

SCPI Commands

```
DISPlay:WINDow<Window>:MTABLE
```

class MtableCls

Mtable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AutoMode

```
# SCPI: DISPlay[:WINDow<n>]:MTABLE
value: enums.AutoMode = driver.applications.k50Spurious.display.window.mtable.
↪get(window = repcap.Window.Default)
```

This command turns the marker table on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

display_mode: ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table.

set(display_mode: AutoMode, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:MTABLE
driver.applications.k50Spurious.display.window.mtable.set(display_mode = enums.
↪AutoMode.AUTO, window = repcap.Window.Default)
```

This command turns the marker table on and off.

param display_mode

ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.4 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.display.window.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.display.window.pmeter.repcap_powerMeter_set(repcap.
↳PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.pmeter.clone()
```

Subgroups

6.1.4.3.2.5 State

SCPI Commands

```
DISPlay:WINDow<Window>:PMETer<PowerMeter>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, powerMeter=PowerMeter.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:PMETer<p>:STATe
value: bool = driver.applications.k50Spurious.display.window.pmeter.state.
↳get(window = repcap.Window.Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: bool, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:PMETer<p>:STATE
driver.applications.k50Spurious.display.window.pmeter.state.set(arg_0 = False,
↳window = repcap.Window.Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param arg_0

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.3.2.6 Size

SCPI Commands

```
DISPlay:WINDow<Window>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → Size

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
value: enums.Size = driver.applications.k50Spurious.display.window.size.
↳get(window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

window_size: No help available

set(window_size: Size, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
driver.applications.k50Spurious.display.window.size.set(window_size = enums.
↳Size.LARGe, window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param window_size

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.7 Trace<Trace>**RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.applications.k50Spurious.display.window.trace.repcap_trace_get()
driver.applications.k50Spurious.display.window.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 9 total commands, 3 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.trace.clone()
```

Subgroups**6.1.4.3.2.8 Length****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:LENGth
value: float = driver.applications.k50Spurious.display.window.trace.length.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return
 trace_length: No help available

6.1.4.3.2.9 State

SCPI Commands

DISPlay:WINDow<Window>:TRACe<Trace>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

<pre># SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe] value: bool = driver.applications.k50Spurious.display.window.trace.state. ↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>
--

No command help available

param window
 optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace
 optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return
 state: No help available

set(state: bool, window=Window.Default, trace=Trace.Default) → None

<pre># SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe] driver.applications.k50Spurious.display.window.trace.state.set(state = False,↳ ↳window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>

No command help available

param state
 No help available

param window
 optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace
 optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.4.3.2.10 Y

class YCls

Y commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.trace.y.clone()
```

Subgroups

6.1.4.3.2.11 Scale

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE
```

class ScaleCls

Scale commands group definition. 7 total commands, 6 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

scale: No help available

set(scale: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]
driver.applications.k50Spurious.display.window.trace.y.scale.set(scale = 1.0,
↳window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param scale

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.display.window.trace.y.scale.clone()
```

Subgroups**6.1.4.3.2.12 Auto****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe:Y:SCALE:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALE]:AUTO
value: bool = driver.applications.k50Spurious.display.window.trace.y.scale.auto.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

auto: No help available

set(auto: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALE]:AUTO
driver.applications.k50Spurious.display.window.trace.y.scale.auto.set(auto =
↳False, window = repcap.Window.Default)
```

No command help available

param auto

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.13 Maximum

SCPI Commands

`DISPlay:WINDow<Window>:TRACe:Y:SCALe:MAXimum`

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:MAXimum
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↳ maximum.get(window = repcap.Window.Default)
```

Defines the maximum value on the y-axis in the specified window.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

max_py: numeric value

set(max_py: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:MAXimum
driver.applications.k50Spurious.display.window.trace.y.scale.maximum.set(max_py,
↳ 1.0, window = repcap.Window.Default)
```

Defines the maximum value on the y-axis in the specified window.

param max_py

numeric value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.14 Minimum

SCPI Commands

`DISPlay:WINDow<Window>:TRACe:Y:SCALe:MINimum`

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:MINimum
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↳ minimum.get(window = repcap.Window.Default)
```

Defines the minimum value on the y-axis in the specified window.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

min_py: numeric value

set(min_py: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:MINimum
driver.applications.k50Spurious.display.window.trace.y.scale.minimum.set(min_py,
↪= 1.0, window = repcap.Window.Default)
```

Defines the minimum value on the y-axis in the specified window.

param min_py

numeric value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.15 Pdivision

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe:Y:SCALe:PDIVision
```

class PdivisionCls

Pdivision commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:PDIVision
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↪pdivision.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

per_division: No help available

set(per_division: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:PDIVision
driver.applications.k50Spurious.display.window.trace.y.scale.pdivision.set(per_
↪division = 1.0, window = repcap.Window.Default)
```

No command help available

param per_division

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.16 RefPosition**SCPI Commands**

`DISPlay:WINDow<Window>:TRACe:Y:SCALe:RPOSition`

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RPOSition
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↪ refPosition.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

position: No help available

set(*position: float, window=Window.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RPOSition
driver.applications.k50Spurious.display.window.trace.y.scale.refPosition.
↪ set(position = 1.0, window = repcap.Window.Default)
```

No command help available

param position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.2.17 Rvalue**SCPI Commands**

`DISPlay:WINDow<Window>:TRACe:Y:SCALe:RVALue`

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RVALue
value: float = driver.applications.k50Spurious.display.window.trace.y.scale.
↪ rvalue.get(window = repcap.Window.Default)
```

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

ref_value: No help available

set(ref_value: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RVALue
driver.applications.k50Spurious.display.window.trace.y.scale.rvalue.set(ref_
↪ value = 1.0, window = repcap.Window.Default)
```

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param ref_value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.4.3.3 Wselect

SCPI Commands

DISPlay:WSElect

class WselectCls

Wselect commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: DISPlay:WSElect
value: int = driver.applications.k50Spurious.display.wselect.get()
```

No command help available

return

selected_window: No help available

6.1.4.4 Fetch

class FetchCls

Fetch commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.fetch.clone()
```

Subgroups

6.1.4.4.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.fetch.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.fetch.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

SCPI Commands

```
FETCh:PMETer<PowerMeter>
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

get(powerMeter=PowerMeter.Default) → List[float]

```
# SCPI: FETCh:PMETer<p>
value: List[float] = driver.applications.k50Spurious.fetch.pmeter.
    ↪get(powerMeter = repcap.PowerMeter.Default)
```

This command queries the results of power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.fetch.pmeter.clone()
```

6.1.4.5 FormatPy

class FormatPyCls

FormatPy commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.formatPy.clone()
```

Subgroups

6.1.4.5.1 Dexport

class DexportCls

Dexport commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.formatPy.dexport.clone()
```

Subgroups

6.1.4.5.1.1 Dseparator

SCPI Commands

```
FORMat:DEXPort:DSEPARATOR
```

class DseparatorCls

Dseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Separator

```
# SCPI: FORMat:DEXPort:DSEPARATOR
value: enums.Separator = driver.applications.k50Spurious.formatPy.dexport.
↳ dseparator.get()
```

This command selects the decimal separator for data exported in ASCII format.

return

separator: POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05.
POINTt Uses a point as decimal separator, e.g. 4.05.

set(separator: Separator) → None

```
# SCPI: FORMat:DEXPort:DSEPARATOR
driver.applications.k50Spurious.formatPy.dexport.dseparator.set(separator =
↳enums.Separator.COMMa)
```

This command selects the decimal separator for data exported in ASCII format.

param separator

POINTt | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05. POINTt Uses
a point as decimal separator, e.g. 4.05.

6.1.4.5.1.2 Header

SCPI Commands

```
FORMat:DEXPort:HEADer
```

class HeaderCls

Header commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: FORMat:DEXPort:HEADer
value: bool = driver.applications.k50Spurious.formatPy.dexport.header.get()
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: FORMat:DEXPort:HEADer
driver.applications.k50Spurious.formatPy.dexport.header.set(state = False)
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

param state

ON | OFF | 0 | 1

6.1.4.5.1.3 Traces

SCPI Commands

FORMat:DEXPort:TRACes

class TracesCls

Traces commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SelectionScope

```
# SCPI: FORMat:DEXPort:TRACes
value: enums.SelectionScope = driver.applications.k50Spurious.formatPy.dexport.
↳traces.get()
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set) .

return

mode: No help available

set(mode: SelectionScope) → None

```
# SCPI: FORMat:DEXPort:TRACes
driver.applications.k50Spurious.formatPy.dexport.traces.set(mode = enums.
↳SelectionScope.ALL)
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set) .

param mode

SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

6.1.4.6 Initiate

class InitiateCls

Initiate commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.initiate.clone()
```

Subgroups

6.1.4.6.1 ConMeas

SCPI Commands

INITiate:CONMeas

class ConMeasCls

ConMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:CONMeas
driver.applications.k50Spurious.initiate.conMeas.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:CONMeas
driver.applications.k50Spurious.initiate.conMeas.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.6.2 Continuous

SCPI Commands

INITiate:CONTinuous

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INITiate:CONTinuous
value: bool = driver.applications.k50Spurious.initiate.continuous.get()
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

return

state: ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

set(state: bool) → None

```
# SCPI: INITiate:CONTinuous
driver.applications.k50Spurious.initiate.continuous.set(state = False)
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

param state

ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

6.1.4.6.3 Immediate

SCPI Commands

```
INITiate:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k50Spurious.initiate.immediate.set()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k50Spurious.initiate.immediate.set_with_opc()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.6.4 Spurious

SCPI Commands

INITiate:SPURious

class SpuriousCls

Spurious commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:SPURious
driver.applications.k50Spurious.initiate.spurious.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SPURious
driver.applications.k50Spurious.initiate.spurious.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.6.5 Sync

SCPI Commands

INITiate:SYNC

class SyncCls

Sync commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:SYNC
driver.applications.k50Spurious.initiate.sync.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SYNC
driver.applications.k50Spurious.initiate.sync.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.7 InputPy

class InputPyCls

InputPy commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.inputPy.clone()
```

Subgroups

6.1.4.7.1 Connector

SCPI Commands

INPut:CONNector

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → InputConnectorC

```
# SCPI: INPut:CONNector
value: enums.InputConnectorC = driver.applications.k50Spurious.inputPy.
↳connector.get()
```

Determines which connector the input for the measurement is taken from.

return

input_connectors: No help available

set(input_connectors: InputConnectorC) → None

```
# SCPI: INPut:CONNector
driver.applications.k50Spurious.inputPy.connector.set(input_connectors = enums.
↳InputConnectorC.RF)
```

Determines which connector the input for the measurement is taken from.

param input_connectors

No help available

6.1.4.7.2 Coupling

SCPI Commands

`INPut:COUPling`

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CouplingTypeA

```
# SCPI: INPut:COUPling
value: enums.CouplingTypeA = driver.applications.k50Spurious.inputPy.coupling.
↳get()
```

This command selects the coupling type of the RF input.

return
coupling_type: AC | DC AC AC coupling DC DC coupling

set(coupling_type: CouplingTypeA) → None

```
# SCPI: INPut:COUPling
driver.applications.k50Spurious.inputPy.coupling.set(coupling_type = enums.
↳CouplingTypeA.AC)
```

This command selects the coupling type of the RF input.

param coupling_type
AC | DC AC AC coupling DC DC coupling

6.1.4.7.3 FilterPy

class FilterPyCls

FilterPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.inputPy.filterPy.clone()
```

Subgroups

6.1.4.7.3.1 Hpass

class HpassCls

Hpass commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.inputPy.filterPy.hpass.clone()
```

Subgroups

6.1.4.7.3.2 State

SCPI Commands

```
INPut:FILTer:HPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTer:HPASs[:STATe]
value: bool = driver.applications.k50Spurious.inputPy.filterPy.hpass.state.get()
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: INPut:FILTer:HPASs[:STATe]
driver.applications.k50Spurious.inputPy.filterPy.hpass.state.set(state = False)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.4.7.3.3 Yig

class YigCls

Yig commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.inputPy.filterPy.yig.clone()
```

Subgroups

6.1.4.7.3.4 State

SCPI Commands

```
INPut:FILTER:YIG:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTER:YIG[:STATE]
value: bool = driver.applications.k50Spurious.inputPy.filterPy.yig.state.get()
```

Enables or disables the YIG filter.

return
state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: INPut:FILTER:YIG[:STATE]
driver.applications.k50Spurious.inputPy.filterPy.yig.state.set(state = False)
```

Enables or disables the YIG filter.

param state
ON | OFF | 0 | 1

6.1.4.7.4 Impedance

SCPI Commands

```
INPut:IMPedance
```

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: INPut:IMPedance
value: int = driver.applications.k50Spurious.inputPy.impedance.get()
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

return
impedance: 50 | 75 Unit: OHM

set(impedance: int) → None

```
# SCPI: INPut:IMPedance
driver.applications.k50Spurious.inputPy.impedance.set(impedance = 1)
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

param impedance
50 | 75 Unit: OHM

6.1.4.8 Layout

class LayoutCls

Layout commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.clone()
```

Subgroups

6.1.4.8.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.add.clone()
```

Subgroups

6.1.4.8.1.1 Window

SCPI Commands

LAYout:ADD:WINDow

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK50) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k50Spurious.layout.add.window.get(window_name_,
↪= '1', direction = enums.WindowDirection.ABOVe, window_type = enums.
↪WindowTypeK50.MarkerTable=MTABle)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method `RsFswp.Layout.Replace.Window.set` command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method `RsFswp.Layout.Catalog.Window.get_query`.

param direction

LEFT | RIGHT | ABOVe | BELow Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.4.8.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.catalog.clone()
```

Subgroups

6.1.4.8.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k50Spurious.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return
result: No help available

6.1.4.8.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.identify.clone()
```

Subgroups

6.1.4.8.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k50Spurious.layout.identify.window.get(window_
↪name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name

String containing the name of a window.

return

window_index: Index number of the window.

6.1.4.8.4 Move

class MoveCls

Move commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.move.clone()
```

Subgroups

6.1.4.8.4.1 Window

SCPI Commands

```
LAYout:MOVE:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(source_window: str, target_window: str, arg_2: WindowDirReplace) → None

```
# SCPI: LAYout:MOVE[:WINDow]
driver.applications.k50Spurious.layout.move.window.set(source_window = '1', ↪
↪target_window = '1', arg_2 = enums.WindowDirReplace.ABOVE)
```

No command help available

param source_window

No help available

param target_window

No help available

param arg_2

No help available

6.1.4.8.5 Remove

class RemoveCls

Remove commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.remove.clone()
```

Subgroups

6.1.4.8.5.1 Window

SCPI Commands

```
LAYout:REMove:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str) → None

```
# SCPI: LAYout:REMove[:WINDow]
driver.applications.k50Spurious.layout.remove.window.set(window_name = '1')
```

This command removes a window from the display in the active channel.

param window_name

String containing the name of the window. In the default state, the name of the window is its index.

6.1.4.8.6 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.layout.replace.clone()
```

Subgroups

6.1.4.8.6.1 Window

SCPI Commands

LAYout:REPLace:WINDow

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeK50) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k50Spurious.layout.replace.window.set(window_name = '1',
↪window_type = enums.WindowTypeK50.MarkerTable=MTABLE)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method `RsFswp.Layout.Add.Window.get_` command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method `RsFswp.Layout.Catalog.Window.get_` query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method `RsFswp.Layout.Add.Window.get_` for a list of available window types.

6.1.4.8.7 Splitter

SCPI Commands

LAYout:SPLitter

class SplitterCls

Splitter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(index_1: int, index_2: int, position: int) → None

```
# SCPI: LAYout:SPLitter
driver.applications.k50Spurious.layout.splitter.set(index_1 = 1, index_2 = 1,
↪position = 1)
```

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method `RsFswp.Applications.K30_NoiseFigure.Display.Window.Size.set` command, the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

param index_1

The index of one window the splitter controls.

param index_2

The index of a window on the other side of the splitter.

param position

New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu) . The point of origin (x = 0, y = 0) is in the lower left corner of the screen. The end point (x = 100, y = 100) is in the upper right corner of the screen. (See Figure ‘SmartGrid coordinates for remote control of the splitters’.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

6.1.4.9 MassMemory**class MassMemoryCls**

MassMemory commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.massMemory.clone()
```

Subgroups**6.1.4.9.1 Store<Store>****RepCap Settings**

```
# Range: Pos1 .. Pos32
rc = driver.applications.k50Spurious.massMemory.store.repcap_store_get()
driver.applications.k50Spurious.massMemory.store.repcap_store_set(repcap.Store.Pos1)
```

class StoreCls

Store commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: Store, default value after init: Store.Pos1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.massMemory.store.clone()
```

Subgroups

6.1.4.9.1.1 Spur

class SpurCls

Spur commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.massMemory.store.spur.clone()
```

Subgroups

6.1.4.9.1.2 Meas

SCPI Commands

```
MMEMory:STORe:SPUR:MEAS
```

class MeasCls

Meas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file: str) → None

```
# SCPI: MMEMory:STORe:SPUR:MEAS
driver.applications.k50Spurious.massMemory.store.spur.meas.set(file = '1')
```

This command stores the current measurement results (all enabled traces and tables of all windows) into the specified csv file. The results are output in the same order as they are displayed on the screen: window by window, trace by trace, and table row by table row.

param file

No help available

6.1.4.9.1.3 Table

SCPI Commands

```
MMEMory:STORe<Store>:TABLE
```

class TableCls

Table commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(columns: StatisticType, filename: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:TABLE
driver.applications.k50Spurious.massMemory.store.table.set(columns = enums.
↳StatisticType.ALL, filename = '1', store = repcap.Store.Default)
```

Exports the selected data from the specified window as a comma-separated list of results, table row by table row, to an ASCII file. The decimal separator (decimal point or comma) for floating-point numerals contained in the file is defined by method `RsFswp.Applications.K30_NoiseFigure.FormatPy.Dexport.Dseparator.set`.

param columns

SElected | ALL Defines which columns to include in the export file. SElected Only the results defined by method `RsFswp.Applications.K50_Spurious.Calculate.Ssearch.Table.Column.set` are included. ALL All available results are included.

param filename

String containing the path and name of the file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.1.4.9.1.4 Trace

SCPI Commands

```
MMEMory:STORe<Store>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*trace: int, filename: str, store=Store.Default*) → None

```
# SCPI: MMEMory:STORe<n>:TRACe
driver.applications.k50Spurious.massMemory.store.trace.set(trace = 1, filename_
↳= '1', store = repcap.Store.Default)
```

This command exports trace data from the specified window to an ASCII file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a 'memory limit reached' error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device.

param trace

Number of the trace to be stored

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.1.4.10 Output

class OutputCls

Output commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.output.clone()
```

Subgroups

6.1.4.10.1 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k50Spurious.output.trigger.repcap_triggerPort_get()
driver.applications.k50Spurious.output.trigger.repcap_triggerPort_set(repcap.TriggerPort.
↪Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 4 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.output.trigger.clone()
```

Subgroups

6.1.4.10.1.1 Direction

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → InOutDirection

```
# SCPI: OUTPut:TRIGger<tp>:DIRection
value: enums.InOutDirection = driver.applications.k50Spurious.output.trigger.
↪direction.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

direction: INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

set(*direction: InOutDirection, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut:TRIGger<tp>:DIRection
driver.applications.k50Spurious.output.trigger.direction.set(direction = enums.
↳ InOutDirection.INPut, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param direction

INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.4.10.1.2 Level

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*triggerPort=TriggerPort.Default*) → LowHigh

```
# SCPI: OUTPut:TRIGger<tp>:LEVel
value: enums.LowHigh = driver.applications.k50Spurious.output.trigger.level.
↳ get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

level: HIGH 5 V LOW 0 V

set(*level: LowHigh, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut:TRIGger<tp>:LEVel
driver.applications.k50Spurious.output.trigger.level.set(level = enums.LowHigh.
↳ HIGH, triggerPort = repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method `RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set`.

param level

HIGH 5 V LOW 0 V

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.4.10.1.3 Otype

SCPI Commands

`OUTPut:TRIGger<TriggerPort>:OTYPE`

class OtypeCls

Otype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*triggerPort*=*TriggerPort.Default*) → *TriggerOutType*

```
# SCPI: OUTPut:TRIGger<tp>:OTYPE
value: enums.TriggerOutType = driver.applications.k50Spurious.output.trigger.
↳ otype.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

type_py: No help available

set(*type_py*: *TriggerOutType*, *triggerPort*=*TriggerPort.Default*) → None

```
# SCPI: OUTPut:TRIGger<tp>:OTYPE
driver.applications.k50Spurious.output.trigger.otype.set(type_py = enums.
↳ TriggerOutType.DEVICE, triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param type_py

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.4.10.1.4 Pulse

class PulseCls

Pulse commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.output.trigger.pulse.clone()
```

Subgroups

6.1.4.10.1.5 Immediate

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:PULSe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:IMMediate
driver.applications.k50Spurious.output.trigger.pulse.immediate.set(triggerPort,
↪= repcap.TriggerPort.Default)
```

This command generates a pulse at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

set_with_opc(triggerPort=TriggerPort.Default, opc_timeout_ms: int = -1) → None

6.1.4.10.1.6 Length

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:PULSe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:LENGth
value: float = driver.applications.k50Spurious.output.trigger.pulse.length.
↪get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

length: Pulse length in seconds. Unit: S

set(length: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:LENGth
driver.applications.k50Spurious.output.trigger.pulse.length.set(length = 1.0,
↪ triggerPort = repcap.TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param length

Pulse length in seconds. Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.4.11 Read

class ReadCls

Read commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.read.clone()
```

Subgroups

6.1.4.11.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.read.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.read.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

SCPI Commands

```
READ:PMETer<PowerMeter>
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

get(powerMeter=PowerMeter.Default) → List[float]

```
# SCPI: READ:PMETer<p>
value: List[float] = driver.applications.k50Spurious.read.pmeter.get(powerMeter=
↳= repcap.PowerMeter.Default)
```

This command initiates a power sensor measurement and queries the results.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.read.pmeter.clone()
```

6.1.4.12 Sense

class SenseCls

Sense commands group definition. 131 total commands, 13 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.clone()
```

Subgroups

6.1.4.12.1 Adjust

class AdjustCls

Adjust commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.adjust.clone()
```

Subgroups

6.1.4.12.1.1 Carrier

SCPI Commands

```
SENSe:ADJust:CARRier
```

class CarrierCls

Carrier commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADJust:CARRier
driver.applications.k50Spurious.sense.adjust.carrier.set()
```

Automatically detects the highest peak over the complete frequency range of the analyzer. This value is considered to be the reference carrier and is indicated in ‘Carrier Level’.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADJust:CARRier
driver.applications.k50Spurious.sense.adjust.carrier.set_with_opc()
```

Automatically detects the highest peak over the complete frequency range of the analyzer. This value is considered to be the reference carrier and is indicated in ‘Carrier Level’.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.12.1.2 Level

SCPI Commands

```
SENSe:ADJust:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k50Spurious.sense.adjust.level.set()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k50Spurious.sense.adjust.level.set_with_opc()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.12.2 Correction

class CorrectionCls

Correction commands group definition. 11 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.correction.clone()
```

Subgroups

6.1.4.12.2.1 Cvl

SCPI Commands

```
SENSe:CORRection:CVL:CLEar
```

class CvlCls

Cvl commands group definition. 11 total commands, 10 Subgroups, 1 group commands

clear() → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEar
driver.applications.k50Spurious.sense.correction.cvl.clear()
```

This command deletes the selected conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

clear_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEar
driver.applications.k50Spurious.sense.correction.cvl.clear_with_opc()
```

This command deletes the selected conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

Same as clear, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.correction.cvl.clone()
```

Subgroups

6.1.4.12.2.2 Band

SCPI Commands

```
SENSe:CORRection:CVL:BAND
```

class BandCls

Band commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → BandB

```
# SCPI: [SENSe]:CORRection:CVL:BAND
value: enums.BandB = driver.applications.k50Spurious.sense.correction.cvl.band.
↳get()
```

This command defines the waveguide band for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return

band: (enum or string) K | KA | Q | U | V | E | W | F | D | G | Y | J | USER Standard waveguide band or user-defined band. For a definition of the frequency range for the pre-defined bands, see Table ‘Frequency ranges for pre-defined bands’).

set(band: BandB) → None

```
# SCPI: [SENSe]:CORRection:CVL:BAND
driver.applications.k50Spurious.sense.correction.cvl.band.set(band = enums.
↳BandB.D)
```


This command defines the waveguide band for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param band

(enum or string) K | KA | Q | U | V | E | W | F | D | G | Y | J | USER Standard waveguide band or user-defined band. For a definition of the frequency range for the pre-defined bands, see Table 'Frequency ranges for pre-defined bands' .

6.1.4.12.2.3 Bias

SCPI Commands

```
SENSe:CORRection:CVL:BIAS
```

class BiasCls

Bias commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
value: float = driver.applications.k50Spurious.sense.correction.cv1.bias.get()
```

This command defines the bias setting to be used with the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

return

bias: No help available

set(bias: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
driver.applications.k50Spurious.sense.correction.cv1.bias.set(bias = 1.0)
```

This command defines the bias setting to be used with the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

param bias

Unit: A

6.1.4.12.2.4 Catalog

SCPI Commands

```
SENSe:CORRection:CVL:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:CATalog
value: float = driver.applications.k50Spurious.sense.correction.cv1.catalog.
↳get()
```

This command queries all available conversion loss tables saved in the C:/R_S/INSTR/USER/cvl/ directory on the instrument. This command is only available with option B21 (External Mixer) installed.

return

catalog: 'string' Comma-separated list of strings containing the file names.

6.1.4.12.2.5 Comment

SCPI Commands

SENSe:CORRection:CVL:COMMeNt

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(comment: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:COMMeNt
value: str = driver.applications.k50Spurious.sense.correction.cv1.comment.
↳get(comment = '1')
```

This command defines a comment for the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SELeCt) . This command is only available with option B21 (External Mixer) installed.

param comment

No help available

return

comment: No help available

set(comment: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:COMMeNt
driver.applications.k50Spurious.sense.correction.cv1.comment.set(comment = '1')
```

This command defines a comment for the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SELeCt) . This command is only available with option B21 (External Mixer) installed.

param comment

No help available

6.1.4.12.2.6 Data

SCPI Commands

```
SENSe:CORRection:CVL:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(freq: List[float], level: List[float]) → List[float]

```
# SCPI: [SENSe]:CORRection:CVL:DATA
value: List[float] = driver.applications.k50Spurious.sense.correction.cv1.data.
    ↪get(freq = [1.1, 2.2, 3.3], level = [1.1, 2.2, 3.3])
```

This command defines the reference values of the selected conversion loss tables. The values are entered as a set of frequency/level pairs. A maximum of 50 frequency/level pairs may be entered. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param freq

The frequencies have to be sent in ascending order. Unit: HZ

param level

Unit: DB

return

freq: The frequencies have to be sent in ascending order. Unit: HZ

set(freq: List[float], level: List[float]) → None

```
# SCPI: [SENSe]:CORRection:CVL:DATA
driver.applications.k50Spurious.sense.correction.cv1.data.set(freq = [1.1, 2.2,
    ↪3.3], level = [1.1, 2.2, 3.3])
```

This command defines the reference values of the selected conversion loss tables. The values are entered as a set of frequency/level pairs. A maximum of 50 frequency/level pairs may be entered. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param freq

The frequencies have to be sent in ascending order. Unit: HZ

param level

Unit: DB

6.1.4.12.2.7 Harmonic

SCPI Commands

```
SENSe:CORRection:CVL:HARMonic
```

class HarmonicCls

Harmonic commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*harmonic: float*) → float

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
value: float = driver.applications.k50Spurious.sense.correction.cv1.harmonic.
↳get(harmonic = 1.0)
```

This command defines the harmonic order for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

param harmonic

Range: 2 to 65

return

harmonic: No help available

set(*harmonic: float*) → None

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
driver.applications.k50Spurious.sense.correction.cv1.harmonic.set(harmonic = 1.
↳0)
```

This command defines the harmonic order for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

param harmonic

Range: 2 to 65

6.1.4.12.2.8 Mixer

SCPI Commands

```
SENSe:CORRection:CVL:MIXer
```

class MixerCls

Mixer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*comment: str*) → str

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
value: str = driver.applications.k50Spurious.sense.correction.cv1.mixer.
↳get(comment = '1')
```

This command defines the mixer name in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param comment

string Name of mixer with a maximum of 16 characters

return

comment: No help available

set(comment: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
driver.applications.k50Spurious.sense.correction.cvl.mixer.set(comment = '1')
```

This command defines the mixer name in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param comment

string Name of mixer with a maximum of 16 characters

6.1.4.12.2.9 Ports

SCPI Commands

```
SENSe:CORRection:CVL:PORTs
```

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(port: float) → float

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
value: float = driver.applications.k50Spurious.sense.correction.cvl.ports.
↳get(port = 1.0)
```

This command defines the mixer type in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param port

2 | 3

return

port: No help available

set(port: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
driver.applications.k50Spurious.sense.correction.cvl.ports.set(port = 1.0)
```

This command defines the mixer type in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param port

2 | 3

6.1.4.12.2.10 Select

SCPI Commands

SENSe:CORRection:CVL:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(select: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:SElect
value: str = driver.applications.k50Spurious.sense.correction.cv1.select.
↳get(select = '1')
```

This command selects the conversion loss table with the specified file name. If <file_name> is not available, a new conversion loss table is created. This command is only available with option B21 (External Mixer) installed.

param select

String containing the path and name of the file.

return

select: No help available

set(select: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:SElect
driver.applications.k50Spurious.sense.correction.cv1.select.set(select = '1')
```

This command selects the conversion loss table with the specified file name. If <file_name> is not available, a new conversion loss table is created. This command is only available with option B21 (External Mixer) installed.

param select

String containing the path and name of the file.

6.1.4.12.2.11 Snumber

SCPI Commands

SENSe:CORRection:CVL:SNUMber

class SnumberCls

Snumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(serial_number: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
value: str = driver.applications.k50Spurious.sense.correction.cv1.snumber.
↳get(serial_number = '1')
```

This command defines the serial number of the mixer for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to

the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param serial_number

Serial number with a maximum of 16 characters

return

serial_number: No help available

set(serial_number: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
driver.applications.k50Spurious.sense.correction.cvl.snumber.set(serial_number,
↪= '1')
```

This command defines the serial number of the mixer for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param serial_number

Serial number with a maximum of 16 characters

6.1.4.12.3 Reference

class ReferenceCls

Reference commands group definition. 14 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.reference.clone()
```

Subgroups

6.1.4.12.3.1 Reference

SCPI Commands

```
SENSe:CREference:FREference
```

class ReferenceCls

Reference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ReferenceMode

```
# SCPI: [SENSe]:CREference:FREference
value: enums.ReferenceMode = driver.applications.k50Spurious.sense.reference.
↪reference.get()
```

No command help available

return
limits: ABSolute | RELative

set(limits: *ReferenceMode*) → None

```
# SCPI: [SENSe]:CREference:FREference
driver.applications.k50Spurious.sense.reference.preference.set(limits = enums.
↳ReferenceMode.ABSolute)
```

No command help available

param limits
ABSolute | RELative

6.1.4.12.3.2 Frequency

SCPI Commands

SENSe:CREference:FREquency

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:FREquency
value: float = driver.applications.k50Spurious.sense.reference.frequency.get()
```

Defines or queries the frequency at which the maximum peak of the signal, that is: the reference carrier, was found.

return
frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:CREference:FREquency
driver.applications.k50Spurious.sense.reference.frequency.set(frequency = 1.0)
```

Defines or queries the frequency at which the maximum peak of the signal, that is: the reference carrier, was found.

param frequency
Unit: HZ

6.1.4.12.3.3 Guard

class GuardCls

Guard commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.reference.guard.clone()
```

Subgroups

6.1.4.12.3.4 Interval

SCPI Commands

```
SENSe:CREference:GUARd:INTERval
```

class IntervalCls

Interval commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:GUARd:INTERval
value: float = driver.applications.k50Spurious.sense.reference.guard.interval.
    ↪get()
```

Defines the guard interval as a span around the reference carrier. This setting is only available for [SENSe:]CREference:GUARd:STATe OFF

return
span: Unit: HZ

set(span: float) → None

```
# SCPI: [SENSe]:CREference:GUARd:INTERval
driver.applications.k50Spurious.sense.reference.guard.interval.set(span = 1.0)
```

Defines the guard interval as a span around the reference carrier. This setting is only available for [SENSe:]CREference:GUARd:STATe OFF

param span
Unit: HZ

6.1.4.12.3.5 State

SCPI Commands

`SENSe:CREference:GUARd:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CREference:GUARd:STATe
value: bool = driver.applications.k50Spurious.sense.creference.guard.state.get()
```

Determines whether the specified guard interval is included in the spur search or not. If the guard interval is not included, the spectrum displays contain gaps at the guard intervals.

return

state: ON | OFF | 0 | 1 OFF | 0 Guard interval is not included ON | 1 Guard interval is included

set(state: bool) → None

```
# SCPI: [SENSe]:CREference:GUARd:STATe
driver.applications.k50Spurious.sense.creference.guard.state.set(state = False)
```

Determines whether the specified guard interval is included in the spur search or not. If the guard interval is not included, the spectrum displays contain gaps at the guard intervals.

param state

ON | OFF | 0 | 1 OFF | 0 Guard interval is not included ON | 1 Guard interval is included

6.1.4.12.3.6 Harmonics

class HarmonicsCls

Harmonics commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.creference.harmonics.clone()
```

Subgroups

6.1.4.12.3.7 Identify

SCPI Commands

`SENSe:CREference:HARMonics:IDENtify`

class IdentifyCls

Identify commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CREference:HARMonics:IDENtify
value: bool = driver.applications.k50Spurious.sense.reference.harmonics.
↳ identify.get()
```

Enables or disables the identification of harmonics of the carrier.

return

state: ON | 1 HARmonics are marked OFF | 0 Harmonics are not marked

set(state: bool) → None

```
# SCPI: [SENSe]:CREference:HARMonics:IDENtify
driver.applications.k50Spurious.sense.reference.harmonics.identify.set(state =
↳ False)
```

Enables or disables the identification of harmonics of the carrier.

param state

ON | 1 HARmonics are marked OFF | 0 Harmonics are not marked

6.1.4.12.3.8 Mnumber**SCPI Commands**

SENSe:CREference:HARMonics:MNUMber

class MnumberCls

Mnumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:HARMonics:MNUMber
value: float = driver.applications.k50Spurious.sense.reference.harmonics.
↳ mnumber.get()
```

Sets the maximum harmonics number to be measured.

return

mharm: No help available

set(mharm: float) → None

```
# SCPI: [SENSe]:CREference:HARMonics:MNUMber
driver.applications.k50Spurious.sense.reference.harmonics.mnumber.set(mharm =
↳ 1.0)
```

Sets the maximum harmonics number to be measured.

param mharm

numeric value

6.1.4.12.3.9 Tolerance

SCPI Commands

`SENSe:CREference:HARMonics:TOLerance`

class ToleranceCls

Tolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:HARMonics:TOLerance
value: float = driver.applications.k50Spurious.sense.creference.harmonics.
↳tolerance.get()
```

Sets the frequency tolerance to match harmonics to measured spurs.

return
tol: No help available

set(tol: float) → None

```
# SCPI: [SENSe]:CREference:HARMonics:TOLerance
driver.applications.k50Spurious.sense.creference.harmonics.tolerance.set(tol =
↳1.0)
```

Sets the frequency tolerance to match harmonics to measured spurs.

param tol
numeric value Unit: Hz

6.1.4.12.3.10 Pdetect

class PdetectCls

Pdetect commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.creference.pdetect.clone()
```

Subgroups

6.1.4.12.3.11 Range

class RangeCls

Range commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.creference.pdetect.range.clone()
```

Subgroups

6.1.4.12.3.12 Center

SCPI Commands

```
SENSe:CREference:PDEtect:RANGe:CENTer
```

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:CENTer
value: float = driver.applications.k50Spurious.sense.creference.pdetect.range.
    ↪ center.get()
```

Defines the center of the range in which the maximum peak is searched.

```
return
    center: Unit: HZ
```

set(center: float) → None

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:CENTer
driver.applications.k50Spurious.sense.creference.pdetect.range.center.
    ↪ set(center = 1.0)
```

Defines the center of the range in which the maximum peak is searched.

```
param center
    Unit: HZ
```

6.1.4.12.3.13 Span

SCPI Commands

```
SENSe:CREference:PDEtect:RANGe:SPAN
```

class SpanCls

Span commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:SPAN
value: float = driver.applications.k50Spurious.sense.creference.pdetect.range.
    ↪ span.get()
```

Defines the width of the range in which the maximum peak is searched.

return
span: Unit: HZ

set(span: float) → None

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:SPAN
driver.applications.k50Spurious.sense.creference.pdetect.range.span.set(span =
↪ 1.0)
```

Defines the width of the range in which the maximum peak is searched.

param span
Unit: HZ

6.1.4.12.3.14 Start

SCPI Commands

SENSe:CREference:PDEtect:RANGe:STARt

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:STARt
value: float = driver.applications.k50Spurious.sense.creference.pdetect.range.
↪ start.get()
```

Defines the beginning of the range in which the maximum peak is searched.

return
start: Unit: HZ

set(start: float) → None

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:STARt
driver.applications.k50Spurious.sense.creference.pdetect.range.start.set(start,
↪ 1.0)
```

Defines the beginning of the range in which the maximum peak is searched.

param start
Unit: HZ

6.1.4.12.3.15 Stop

SCPI Commands

```
SENSe:CREference:PDEtect:RANGe:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:STOP
value: float = driver.applications.k50Spurious.sense.creference.pdetect.range.
↳ stop.get()
```

Defines the end of the range in which the maximum peak is searched.

return
stop: Unit: HZ

set(stop: float) → None

```
# SCPI: [SENSe]:CREference:PDEtect:RANGe:STOP
driver.applications.k50Spurious.sense.creference.pdetect.range.stop.set(stop =
↳ 1.0)
```

Defines the end of the range in which the maximum peak is searched.

param stop
Unit: HZ

6.1.4.12.3.16 Preference

SCPI Commands

```
SENSe:CREference:PREference
```

class PreferenceCls

Preference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ReferenceMode

```
# SCPI: [SENSe]:CREference:PREference
value: enums.ReferenceMode = driver.applications.k50Spurious.sense.creference.
↳ preference.get()
```

No command help available

return
limits: ABSolute | RELative

set(limits: ReferenceMode) → None

```
# SCPI: [SENSe]:CREference:PREference
driver.applications.k50Spurious.sense.creference.preference.set(limits = enums.
↳ReferenceMode.ABSolute)
```

No command help available

param limits
ABSolute | RELative

6.1.4.12.3.17 Srange

SCPI Commands

```
SENSe:CREference:SRANGE
```

class SrangeCls

Srange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SearchRange

```
# SCPI: [SENSe]:CREference:SRANGE
value: enums.SearchRange = driver.applications.k50Spurious.sense.creference.
↳srange.get()
```

Determines the search area for the automatic carrier measurement function.

return
search_range: GMAXimum | RMAXimum GMAXimum Global maximum: The maximum peak in the entire measurement span is determined. RMAXimum Range maximum: The maximum peak is searched only in the specified range.

set(search_range: SearchRange) → None

```
# SCPI: [SENSe]:CREference:SRANGE
driver.applications.k50Spurious.sense.creference.srange.set(search_range =
↳enums.SearchRange.GMAXimum)
```

Determines the search area for the automatic carrier measurement function.

param search_range
GMAXimum | RMAXimum GMAXimum Global maximum: The maximum peak in the entire measurement span is determined. RMAXimum Range maximum: The maximum peak is searched only in the specified range.

6.1.4.12.3.18 Value

SCPI Commands

```
SENSe:CREference:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CREference:VALue
value: float = driver.applications.k50Spurious.sense.reference.value.get()
```

Defines the maximum peak of the signal, which is considered to be the reference carrier.

```
return
    max_peak: Unit: DBM
```

set(max_peak: float) → None

```
# SCPI: [SENSe]:CREference:VALue
driver.applications.k50Spurious.sense.reference.value.set(max_peak = 1.0)
```

Defines the maximum peak of the signal, which is considered to be the reference carrier.

```
param max_peak
    Unit: DBM
```

6.1.4.12.4 Directed

SCPI Commands

```
SENSe:DIReCted:SAVE
SENSe:DIReCted:LOAD
```

class DirectedCls

Directed commands group definition. 11 total commands, 7 Subgroups, 2 group commands

load(filename: str) → None

```
# SCPI: [SENSe]:DIReCted:LOAD
driver.applications.k50Spurious.sense.directed.load(filename = '1')
```

Loads a stored search configuration from a .csv file. The current settings in the table are overwritten by the settings in the file!

```
param filename
    No help available
```

save(filename: str) → None

```
# SCPI: [SENSe]:DIReCted:SAVE
driver.applications.k50Spurious.sense.directed.save(filename = '1')
```

Saves the current directed search configuration to a user-defined .csv file for later use. The result is a comma-separated list of values with the following syntax for each span: <No>,<Frequency>,<SearchSpan>, <Det-Threshold>,<SNR>,<DetectMode> For details on the parameters see ‘Directed Search Measurement settings’).

```
param filename
    No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.directed.clone()
```

Subgroups

6.1.4.12.4.1 InputPy

class InputPyCls

InputPy commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.directed.inputPy.clone()
```

Subgroups

6.1.4.12.4.2 Attenuation

SCPI Commands

```
SENSe:DIReCted:INPut:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIReCted:INPut:ATTenuation
value: float = driver.applications.k50Spurious.sense.directed.inputPy.
↳attenuation.get()
```

Defines the RF attenuation for the directed search measurement.

return

attenuation: integer Range: 0 dB to 79 dB, Unit: DB

set(attenuation: float) → None

```
# SCPI: [SENSe]:DIReCted:INPut:ATTenuation
driver.applications.k50Spurious.sense.directed.inputPy.attenuation.
↳set(attenuation = 1.0)
```

Defines the RF attenuation for the directed search measurement.

param attenuation

integer Range: 0 dB to 79 dB, Unit: DB

6.1.4.12.4.3 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.directed.inputPy.gain.clone()
```

Subgroups

6.1.4.12.4.4 State

SCPI Commands

```
SENSe:DIReCted:INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DIReCted:INPut:GAIN:STATe
value: bool = driver.applications.k50Spurious.sense.directed.inputPy.gain.state.
↳get()
```

Switches the optional preamplifier on or off (if available) for the directed search measurement.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DIReCted:INPut:GAIN:STATe
driver.applications.k50Spurious.sense.directed.inputPy.gain.state.set(state =↳
↳False)
```

Switches the optional preamplifier on or off (if available) for the directed search measurement.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.4.12.4.5 Value

SCPI Commands

```
SENSe:DIReCted:INPut:GAIN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIReCted:INPut:GAIN[:VALue]
value: float = driver.applications.k50Spurious.sense.directed.inputPy.gain.
↳value.get()
```

Defines the gain by the optional preamplifier (if activated for the directed search measurement, see [SENSe:]DIReCted:INPut:GAIN:STATe) . For R&S FSWP26 or higher models, the input signal is amplified by 30 dB if the preamplifier is activated. For R&S FSWP8 or R&S FSWP13 models, different settings are available.

return

gain: 15 dB | 30 dB All other values are rounded to the nearest of these two.

set(gain: float) → None

```
# SCPI: [SENSe]:DIReCted:INPut:GAIN[:VALue]
driver.applications.k50Spurious.sense.directed.inputPy.gain.value.set(gain = 1.
↳0)
```

Defines the gain by the optional preamplifier (if activated for the directed search measurement, see [SENSe:]DIReCted:INPut:GAIN:STATe) . For R&S FSWP26 or higher models, the input signal is amplified by 30 dB if the preamplifier is activated. For R&S FSWP8 or R&S FSWP13 models, different settings are available.

param gain

15 dB | 30 dB All other values are rounded to the nearest of these two.

6.1.4.12.4.6 Loffset

SCPI Commands

```
SENSe:DIReCted:LOFFset
```

class LoffsetCls

Loffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIReCted:LOFFset
value: float = driver.applications.k50Spurious.sense.directed.loffset.get()
```

Defines a limit line as an offset to the detection threshold for each range.

return

peak_exc: Range: 0 to 200, Unit: DB

set(*peak_exc*: float) → None

```
# SCPI: [SENSe]:DIReCted:LOFFset
driver.applications.k50Spurious.sense.directed.loffset.set(peak_exc = 1.0)
```

Defines a limit line as an offset to the detection threshold for each range.

param peak_exc
Range: 0 to 200, Unit: DB

6.1.4.12.4.7 MfRbw

SCPI Commands

```
SENSe:DIReCted:MFRBw
```

class MfRbwCls

MfRbw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIReCted:MFRBw
value: float = driver.applications.k50Spurious.sense.directed.mfRbw.get()
```

No command help available

return
max_final_rbw: Range: 1 Hz to 10 MHz, Unit: HZ

set(*max_final_rbw*: float) → None

```
# SCPI: [SENSe]:DIReCted:MFRBw
driver.applications.k50Spurious.sense.directed.mfRbw.set(max_final_rbw = 1.0)
```

No command help available

param max_final_rbw
Range: 1 Hz to 10 MHz, Unit: HZ

6.1.4.12.4.8 Nfft

SCPI Commands

```
SENSe:DIReCted:NFFT
```

class NfftCls

Nfft commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIReCted:NFFT
value: float = driver.applications.k50Spurious.sense.directed.nfft.get()
```

Defines the number of FFTs to be performed for all spurs in the directed search measurement.

return
loffset: integer Range: 1 to 20, Unit: DB

set(loffset: float) → None

```
# SCPI: [SENSe]:DIRected:NFFT
driver.applications.k50Spurious.sense.directed.nfft.set(loffset = 1.0)
```

Defines the number of FFTs to be performed for all spurs in the directed search measurement.

param loffset
integer Range: 1 to 20, Unit: DB

6.1.4.12.4.9 Pexcursion

SCPI Commands

```
SENSe:DIRected:PEXCursion
```

class PexcursionCls

Pexcursion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIRected:PEXCursion
value: float = driver.applications.k50Spurious.sense.directed.pexcursion.get()
```

Defines the minimum level value by which the signal must rise or fall after a detected spur so that a new spur is detected.

return
peak_exc: Range: 0 to 100, Unit: DB

set(peak_exc: float) → None

```
# SCPI: [SENSe]:DIRected:PEXCursion
driver.applications.k50Spurious.sense.directed.pexcursion.set(peak_exc = 1.0)
```

Defines the minimum level value by which the signal must rise or fall after a detected spur so that a new spur is detected.

param peak_exc
Range: 0 to 100, Unit: DB

6.1.4.12.4.10 RefLevel

SCPI Commands

```
SENSe:DIRected:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DIRected:RLEVel
value: float = driver.applications.k50Spurious.sense.directed.refLevel.get()
```

Defines the reference level for the directed search measurement.

return

ref_level: (-10 dBm + RF attenuation – RF preamplifier gain) Range: -130 dBm to max. 30 dBm, Unit: dBm

set(ref_level: float) → None

```
# SCPI: [SENSe]:DIRected:RLEVel
driver.applications.k50Spurious.sense.directed.refLevel.set(ref_level = 1.0)
```

Defines the reference level for the directed search measurement.

param ref_level

(-10 dBm + RF attenuation – RF preamplifier gain) Range: -130 dBm to max. 30 dBm, Unit: dBm

6.1.4.12.4.11 Settings

SCPI Commands

```
SENSe:DIRected:SETTings
```

class SettingsCls

Settings commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Search_Span: List[float]: numeric value The span around the frequency for which a detailed measurement (spurious detection sweep and spot search) is performed. Note that the frequency spans must be distinct, that is: they may not overlap. Unit: HZ
- Det_Threshold: List[float]: numeric value Absolute threshold that the power level must exceed for a spur to be detected. Unit: dBm
- Desired_Spur_Snr: List[float]: numeric value Minimum signal-to-noise ratio that the power level must exceed for a spur to be detected during the spot search Unit: dB

get() → GetStruct

```
# SCPI: [SENSe]:DIRected:SETTings
value: GetStruct = driver.applications.k50Spurious.sense.directed.settings.get()
```

Defines the current directed search configuration, that is: all frequency spans to be measured in detail. The current configuration table is overwritten. Note that all entries must be defined in one command so that the R&S FSWP Spurious measurements application can detect any possible conflicts between the frequency spans. The parameters are defined as a comma-separated list with one line per span, using the following syntax: <Frequency>,<SearchSpan>,<DetThreshold>,<SNR> For details on the parameters see 'Directed Search Measurement settings'.

return

structure: for return value, see the help for GetStruct structure arguments.

set(frequency: *Optional[List[float]] = None*, search_span: *Optional[List[float]] = None*, det_threshold: *Optional[List[float]] = None*, desired_spur_snr: *Optional[List[float]] = None*) → None

```
# SCPI: [SENSe]:DIReCted:SETTings
driver.applications.k50Spurious.sense.directed.settings.set(frequency = [1.1, 2.
↪2, 3.3], search_span = [1.1, 2.2, 3.3], det_threshold = [1.1, 2.2, 3.3],
↪desired_spur_snr = [1.1, 2.2, 3.3])
```

Defines the current directed search configuration, that is: all frequency spans to be measured in detail. The current configuration table is overwritten. Note that all entries must be defined in one command so that the R&S FSWP Spurious measurements application can detect any possible conflicts between the frequency spans. The parameters are defined as a comma-separated list with one line per span, using the following syntax: <Frequency>,<SearchSpan>,<DetThreshold>,<SNR> For details on the parameters see ‘Directed Search Measurement settings’.

param frequency

numeric value Center frequency for directed search measurement of the spur Unit: HZ

param search_span

numeric value The span around the frequency for which a detailed measurement (spurious detection sweep and spot search) is performed. Note that the frequency spans must be distinct, that is: they may not overlap. Unit: HZ

param det_threshold

numeric value Absolute threshold that the power level must exceed for a spur to be detected. Unit: dBm

param desired_spur_snr

numeric value Minimum signal-to-noise ratio that the power level must exceed for a spur to be detected during the spot search Unit: dB

6.1.4.12.5 Fplan

SCPI Commands

```
SENSe:FPLan:SAVE
SENSe:FPLan:LOAD
```

class FplanCls

Fplan commands group definition. 13 total commands, 3 Subgroups, 2 group commands

load(filename: *str*) → None

```
# SCPI: [SENSe]:FPLan:LOAD
driver.applications.k50Spurious.sense.fplan.load(filename = '1')
```

Loads a stored frequency plan configuration from a .csv file.

param filename

No help available

save(filename: str) → None

```
# SCPI: [SENSe]:FPLan:SAVE
driver.applications.k50Spurious.sense.fplan.save(filename = '1')
```

Saves the current frequency plan configuration to a user-defined .csv file for later use. The result is a comma-separated list of values with the following syntax for each row of the frequency plan: <Num>,<Comp>,<InFreq1>,<MaxHarm1>,<InFreq2>,<Fact>,<MaxHarm2>,<Ident2>,<BandCtr>,<BandSpn>

param filename
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.fplan.clone()
```

Subgroups

6.1.4.12.5.1 Component<Component>

RepCap Settings

```
# Range: Ix1 .. Ix32
rc = driver.applications.k50Spurious.sense.fplan.component.repcap_component_get()
driver.applications.k50Spurious.sense.fplan.component.repcap_component_set(repcap.
↪Component.Ix1)
```

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:DELeTe
```

class ComponentCls

Component commands group definition. 9 total commands, 7 Subgroups, 1 group commands Repeated Capability: Component, default value after init: Component.Ix1

delete(component=Component.Default) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:DELeTe
driver.applications.k50Spurious.sense.fplan.component.delete(component = repcap.
↪Component.Default)
```

This command will delete the selected row from the frequency plan.

param component
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

delete_with_opc(component=Component.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.fplan.component.clone()
```

Subgroups

6.1.4.12.5.2 Add

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:ADD
```

class AddCls

Add commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*component=Component.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:ADD
driver.applications.k50Spurious.sense.fplan.component.add.set(component =
↳repcap.Component.Default)
```

Adds a new component below the selected row <co> in the frequency plan. If the command is executed on a row that does not yet exist, this row and all that are missing up to this row are created.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

set_with_opc(*component=Component.Default, opc_timeout_ms: int = -1*) → None

6.1.4.12.5.3 Bcenter

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:BCENter
```

class BcenterCls

Bcenter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default*) → float

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:BCENter
value: float = driver.applications.k50Spurious.sense.fplan.component.bcenter.
↳get(component = repcap.Component.Default)
```

Defines the center of the search span that is evaluated for spur identification within the frequency plan. By default, the defined center frequency is used.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

return
center_freq: Unit: HZ

set(center_freq: float, component=Component.Default) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:BCENter
driver.applications.k50Spurious.sense.fplan.component.bcenter.set(center_freq = 1.0, component = repcap.Component.Default)
```

Defines the center of the search span that is evaluated for spur identification within the frequency plan. By default, the defined center frequency is used.

param center_freq
Unit: HZ

param component
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

6.1.4.12.5.4 Bspan

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:BSPan
```

class BspanCls

Bspan commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(component=Component.Default) → float

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:BSPan
value: float = driver.applications.k50Spurious.sense.fplan.component.bspan.get(component = repcap.Component.Default)
```

Defines the span that is evaluated for spur identification within the frequency plan. By default, the full measurement span is used.

param component
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

return
span: Unit: HZ

set(span: float, component=Component.Default) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:BSPan
driver.applications.k50Spurious.sense.fplan.component.bspan.set(span = 1.0, component = repcap.Component.Default)
```

Defines the span that is evaluated for spur identification within the frequency plan. By default, the full measurement span is used.

param span
Unit: HZ

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

6.1.4.12.5.5 Factor**SCPI Commands**

`SENSe:FPLan:COMPonent<Component>:FACTOR`

class FactorCls

Factor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default*) → int

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:FACTOR
value: int = driver.applications.k50Spurious.sense.fplan.component.factor.
↳get(component = repcap.Component.Default)
```

No command help available

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

return

factor: No help available

set(*factor: int, component=Component.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:FACTOR
driver.applications.k50Spurious.sense.fplan.component.factor.set(factor = 1,↳
↳component = repcap.Component.Default)
```

No command help available

param factor

No help available

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

6.1.4.12.5.6 Identity**SCPI Commands**

`SENSe:FPLan:COMPonent<Component>:IDENtity`

class IdentityCls

Identity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default*) → MixerIdentifier

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:IDENtity
value: enums.MixerIdentifier = driver.applications.k50Spurious.sense.fplan.
↪ component.identity.get(component = repcap.Component.Default)
```

Selects the identifier for the second input frequency for mixers.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

return

type_py: LO | CLOCK

set(*type_py: MixerIdentifier, component=Component.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:IDENtity
driver.applications.k50Spurious.sense.fplan.component.identity.set(type_py =
↪ enums.MixerIdentifier.CLOCK, component = repcap.Component.Default)
```

Selects the identifier for the second input frequency for mixers.

param type_py

LO | CLOCK

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

6.1.4.12.5.7 Port<Port>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.applications.k50Spurious.sense.fplan.component.port.repcap_port_get()
driver.applications.k50Spurious.sense.fplan.component.port.repcap_port_set(repcap.Port.
↪ Nr1)
```

class PortCls

Port commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: Port, default value after init: Port.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.fplan.component.port.clone()
```

Subgroups

6.1.4.12.5.8 Frequency

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:PORT<Port>:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default, port=Port.Default*) → float

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:PORT<1|2>:FREQuency
value: float = driver.applications.k50Spurious.sense.fplan.component.port.
↪ frequency.get(component = repcap.Component.Default, port = repcap.Port.
↪ Default)
```

Defines the frequency of the input signal. For all components after the first one, the output frequency of the previous component is used as the input frequency. For details see ‘Frequency plan and spur identification’.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Component’)

param port

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Port’)

return

frequency: Unit: HZ

set(*frequency: float, component=Component.Default, port=Port.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:PORT<1|2>:FREQuency
driver.applications.k50Spurious.sense.fplan.component.port.frequency.
↪ set(frequency = 1.0, component = repcap.Component.Default, port = repcap.Port.
↪ Default)
```

Defines the frequency of the input signal. For all components after the first one, the output frequency of the previous component is used as the input frequency. For details see ‘Frequency plan and spur identification’.

param frequency

Unit: HZ

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Component’)

param port

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Port’)

6.1.4.12.5.9 Mharmonic

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:PORT<Port>:MHARmonic
```

class MharmonicCls

Mharmonic commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default, port=Port.Default*) → int

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:PORT<1|2>:MHARmonic
value: int = driver.applications.k50Spurious.sense.fplan.component.port.
↳ mharmonic.get(component = repcap.Component.Default, port = repcap.Port.
↳ Default)
```

Defines the maximum harmonic of each input frequency to be considered in calculating mixer products for spur identification. For details see 'Frequency plan and spur identification'.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

param port

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Port')

return

harmonic: Range: 1 to 5

set(*harmonic: int, component=Component.Default, port=Port.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:PORT<1|2>:MHARmonic
driver.applications.k50Spurious.sense.fplan.component.port.mharmonic.
↳ set(harmonic = 1, component = repcap.Component.Default, port = repcap.Port.
↳ Default)
```

Defines the maximum harmonic of each input frequency to be considered in calculating mixer products for spur identification. For details see 'Frequency plan and spur identification'.

param harmonic

Range: 1 to 5

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Component')

param port

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Port')

6.1.4.12.5.10 TypePy

SCPI Commands

```
SENSe:FPLan:COMPonent<Component>:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*component=Component.Default*) → ComponentType

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:TYPE
value: enums.ComponentType = driver.applications.k50Spurious.sense.fplan.
↳ component.typePy.get(component = repcap.Component.Default)
```

Defines the type of component in the signal path. Depending on the type of component, different parameters are available. For details see ‘Frequency plan and spur identification’.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Component’)

return

type_py: MIXer | AMPLifier | MULTiplier | DIVider MIXer Mixes the input signal (RF input or the output of the previous component) with a second input frequency. AMPLifier Amplifies the input signal (RF input or the output of the previous component) . MULTiplier Multiplies the input signal (RF input or the output of the previous component) by a configurable factor n. DIVider Divides the input signal (RF input or the output of the previous component) by a configurable factor n.

set(*type_py: ComponentType, component=Component.Default*) → None

```
# SCPI: [SENSe]:FPLan:COMPonent<co>:TYPE
driver.applications.k50Spurious.sense.fplan.component.typePy.set(type_py =
↳ enums.ComponentType.AMPLifier, component = repcap.Component.Default)
```

Defines the type of component in the signal path. Depending on the type of component, different parameters are available. For details see ‘Frequency plan and spur identification’.

param type_py

MIXer | AMPLifier | MULTiplier | DIVider MIXer Mixes the input signal (RF input or the output of the previous component) with a second input frequency. AMPLifier Amplifies the input signal (RF input or the output of the previous component) . MULTiplier Multiplies the input signal (RF input or the output of the previous component) by a configurable factor n. DIVider Divides the input signal (RF input or the output of the previous component) by a configurable factor n.

param component

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Component’)

6.1.4.12.5.11 Predicted

SCPI Commands

```
SENSe:FPLan:PREDicted:EXPort
```

class PredictedCls

Predicted commands group definition. 1 total commands, 0 Subgroups, 1 group commands

export(filename: str) → None

```
# SCPI: [SENSe]:FPLan:PREDicted:EXPort
driver.applications.k50Spurious.sense.fplan.predicted.export(filename = '1')
```

Saves the current predicted list to a .csv file.

param filename
No help available

6.1.4.12.5.12 Transfer

SCPI Commands

```
SENSe:FPLan:TRANsfer
```

class TransferCls

Transfer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:FPLan:TRANsfer
driver.applications.k50Spurious.sense.fplan.transfer.set()
```

This command will transfer all frequencies that result out of the current frequency plan settings to the directed search settings. For details see ‘Frequency plan and spur identification’.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:FPLan:TRANsfer
driver.applications.k50Spurious.sense.fplan.transfer.set_with_opc()
```

This command will transfer all frequencies that result out of the current frequency plan settings to the directed search settings. For details see ‘Frequency plan and spur identification’.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.1.4.12.6 Frequency

class FrequencyCls

Frequency commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.frequency.clone()
```

Subgroups

6.1.4.12.6.1 Offset

SCPI Commands

```
SENSe:FREquency:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREquency:OFFSet
value: float = driver.applications.k50Spurious.sense.frequency.offset.get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: [SENSe]:FREquency:OFFSet
driver.applications.k50Spurious.sense.frequency.offset.set(frequency = 1.0)
```

No command help available

param frequency

No help available

6.1.4.12.7 ListPy

SCPI Commands

```
SENSe:LIST:SAVE
SENSe:LIST:LOAD
SENSe:LIST:CLEar
```

class ListPyCls

ListPy commands group definition. 22 total commands, 1 Subgroups, 3 group commands

clear() → None

```
# SCPI: [SENSe]:LIST:CLEar
driver.applications.k50Spurious.sense.listPy.clear()
```

Removes all but the first range from the wide search settings table.

clear_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:LIST:CLEar
driver.applications.k50Spurious.sense.listPy.clear_with_opc()
```

Removes all but the first range from the wide search settings table.

Same as clear, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

load(filename: str) → None

```
# SCPI: [SENSe]:LIST:LOAD
driver.applications.k50Spurious.sense.listPy.load(filename = '1')
```

Loads a stored range setup from a .csv file. The current settings in the table are overwritten by the settings in the file!

param filename

No help available

save(filename: str) → None

```
# SCPI: [SENSe]:LIST:SAVE
driver.applications.k50Spurious.sense.listPy.save(filename = '1')
```

Saves the current range setup to a user-defined comma-separated (.csv) file for later use. The values are stored in the following order for each range: <No>,<Start>,<Stop>,<TNRStart>,<TNRStop>,<LimitOffset>,<PeakExcursion>,<SNR>,<AutoRBW>,<RBW>,<MaxFinalRBW>,<Detector>,<DetLength>,<Reserved>,<RefLevel>,<RFAttenuation>,<Preampl>

param filename

String containing the path and name of the file.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.clone()
```

Subgroups

6.1.4.12.7.1 Range<RangePy>

RepCap Settings

```
# Range: Ix1 .. Ix64
rc = driver.applications.k50Spurious.sense.listPy.range.repcap_rangePy_get()
driver.applications.k50Spurious.sense.listPy.range.repcap_rangePy_set(repcap.RangePy.Ix1)
```

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:DELeTe
```

class RangeCls

Range commands group definition. 19 total commands, 13 Subgroups, 1 group commands Repeated Capability: RangePy, default value after init: RangePy.Ix1

delete(rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:DELeTe
driver.applications.k50Spurious.sense.listPy.range.delete(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

delete_with_opc(rangePy=RangePy.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.clone()
```

Subgroups

6.1.4.12.7.2 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.bandwidth.clone()
```

Subgroups

6.1.4.12.7.3 Auto

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:BANDwidth:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(state: bool, rangePy=RangePy.Default) → bool

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANDwidth:AUTO
value: bool = driver.applications.k50Spurious.sense.listPy.range.bandwidth.auto.
↳get(state = False, rangePy = repcap.RangePy.Default)
```

Activates or deactivates automatic definition of the RBW for individual ranges. If necessary, the range is divided further into segments.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANDwidth:AUTO
driver.applications.k50Spurious.sense.listPy.range.bandwidth.auto.set(state =
↳False, rangePy = repcap.RangePy.Default)
```

Activates or deactivates automatic definition of the RBW for individual ranges. If necessary, the range is divided further into segments.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.4 Resolution

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:BANdwidth:RESolution`

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANdwidth[:RESolution]
value: float = driver.applications.k50Spurious.sense.listPy.range.bandwidth.
↳ resolution.get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

rbw: Range: 1 Hz to 10 MHz , Unit: HZ

set(rbw: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANdwidth[:RESolution]
driver.applications.k50Spurious.sense.listPy.range.bandwidth.resolution.set(rbw,
↳ 1.0, rangePy = repcap.RangePy.Default)
```

No command help available

param rbw

Range: 1 Hz to 10 MHz , Unit: HZ

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.5 Count

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:COUNt`

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → int

```
# SCPI: [SENSe]:LIST:RANGe<ri>:COUNt
value: int = driver.applications.k50Spurious.sense.listPy.range.count.
↳ get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

result: No help available

6.1.4.12.7.6 Frequency**class FrequencyCls**

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.frequency.clone()
```

Subgroups**6.1.4.12.7.7 Start****SCPI Commands**

```
SENSe:LIST:RANGe<RangePy>:FREQuency:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:START
value: float = driver.applications.k50Spurious.sense.listPy.range.frequency.
↪start.get(rangePy = repcap.RangePy.Default)
```

This command defines the start frequency of a wide search measurement range. Subsequent ranges must be defined in ascending order of frequencies; however, gaps between ranges are possible.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

start: Range: 0 to max. frequency, Unit: HZ

set(start: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:START
driver.applications.k50Spurious.sense.listPy.range.frequency.start.set(start =
↪1.0, rangePy = repcap.RangePy.Default)
```

This command defines the start frequency of a wide search measurement range. Subsequent ranges must be defined in ascending order of frequencies; however, gaps between ranges are possible.

param start

Range: 0 to max. frequency , Unit: HZ

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.8 Stop**SCPI Commands**

`SENSe:LIST:RANGe<RangePy>:FREQuency:STOP`

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STOP
value: float = driver.applications.k50Spurious.sense.listPy.range.frequency.
↳ stop.get(rangePy = repcap.RangePy.Default)
```

This command defines the stop frequency of a wide search measurement range. The stop frequency must be higher than the start frequency for the same range.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

stop: Range: 0 to max. frequency , Unit: HZ

set(stop: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STOP
driver.applications.k50Spurious.sense.listPy.range.frequency.stop.set(stop = 1.
↳ 0, rangePy = repcap.RangePy.Default)
```

This command defines the stop frequency of a wide search measurement range. The stop frequency must be higher than the start frequency for the same range.

param stop

Range: 0 to max. frequency , Unit: HZ

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.9 InputPy

class InputPyCls

InputPy commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.inputPy.clone()
```

Subgroups

6.1.4.12.7.10 Attenuation

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:ATTenuation
value: float = driver.applications.k50Spurious.sense.listPy.range.inputPy.
↳attenuation.get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

attenuation: Range: 0 dB to 79 dB , Unit: DB

set(attenuation: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:ATTenuation
driver.applications.k50Spurious.sense.listPy.range.inputPy.attenuation.
↳set(attenuation = 1.0, rangePy = repcap.RangePy.Default)
```

No command help available

param attenuation

Range: 0 dB to 79 dB , Unit: DB

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.11 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.inputPy.gain.clone()
```

Subgroups

6.1.4.12.7.12 State

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → bool

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN:STATe
value: bool = driver.applications.k50Spurious.sense.listPy.range.inputPy.gain.
↪ state.get(rangePy = repcap.RangePy.Default)
```

Switches the optional preamplifier on or off (if available) .

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the preamplifier off ON | 1 Switches the preamplifier on

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN:STATe
driver.applications.k50Spurious.sense.listPy.range.inputPy.gain.state.set(state,
↪ False, rangePy = repcap.RangePy.Default)
```

Switches the optional preamplifier on or off (if available) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the preamplifier off ON | 1 Switches the preamplifier on

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.13 Value

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:GAIN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN[:VALue]
value: float = driver.applications.k50Spurious.sense.listPy.range.inputPy.gain.
↳ value.get(rangePy = repcap.RangePy.Default)
```

Defines the value of the optional preamplifier (for [SENSe:]LIST:RANGe<ri>:INPut:GAIN:STATeON) .
For R&S FSWP26 or higher models, the input signal is amplified by 30 dB if the preamplifier is activated.
For R&S FSWP8 or R&S FSWP13 models, the following settings are available:

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

gain: all values other than 15 dB or 30 dB are rounded to the nearest of the two 15 dB
The input signal is amplified by about 15 dB. 30 dB The input signal is amplified by about 30 dB.

set(gain: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN[:VALue]
driver.applications.k50Spurious.sense.listPy.range.inputPy.gain.value.set(gain,
↳ 1.0, rangePy = repcap.RangePy.Default)
```

Defines the value of the optional preamplifier (for [SENSe:]LIST:RANGe<ri>:INPut:GAIN:STATeON) .
For R&S FSWP26 or higher models, the input signal is amplified by 30 dB if the preamplifier is activated.
For R&S FSWP8 or R&S FSWP13 models, the following settings are available:

param gain

all values other than 15 dB or 30 dB are rounded to the nearest of the two 15 dB
The input signal is amplified by about 15 dB. 30 dB The input signal is amplified by about 30 dB.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.14 Insert

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:INSert`

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*direction: LeftRightDirection, rangePy=RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INSert
driver.applications.k50Spurious.sense.listPy.range.insert.set(direction = enums.
↳LeftRightDirection.LEFT, rangePy = repcap.RangePy.Default)
```

Adds a range right or left to the selected one. If the command is used on a range that does not yet exist, the range and all with lower indices up to this one are created.

param direction
LEFT | RIGHT

param rangePy
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.15 Loffset

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:LOFFset`

class LoffsetCls

Loffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy=RangePy.Default*) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LOFFset
value: float = driver.applications.k50Spurious.sense.listPy.range.loffset.
↳get(rangePy = repcap.RangePy.Default)
```

Defines a limit line as an offset to the detection threshold for each range.

param rangePy
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return
loffset: Range: 0 to 20, Unit: DB

set(*loffset: float, rangePy=RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LOFFset
driver.applications.k50Spurious.sense.listPy.range.loffset.set(loffset = 1.0,
↳rangePy = repcap.RangePy.Default)
```

Defines a limit line as an offset to the detection threshold for each range.

param loffset

Range: 0 to 20, Unit: DB

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.16 MfRbw

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:MFRBw
```

class MfRbwCls

MfRbw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:MFRBw
value: float = driver.applications.k50Spurious.sense.listPy.range.mfRbw.
↳ get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

max_final_rbw: Range: 1 Hz to 10 MHz, Unit: HZ

set(max_final_rbw: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:MFRBw
driver.applications.k50Spurious.sense.listPy.range.mfRbw.set(max_final_rbw = 1.
↳ 0, rangePy = repcap.RangePy.Default)
```

No command help available

param max_final_rbw

Range: 1 Hz to 10 MHz, Unit: HZ

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.17 Nfft

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:NFFT
```

class NfftCls

Nfft commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:NFFT
value: float = driver.applications.k50Spurious.sense.listPy.range.nfft.
↳ get(rangePy = repcap.RangePy.Default)
```

Defines the number of FFT averages to be performed for each range or segment.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

loffset: integer Range: 1 to 20, Unit: DB

set(loffset: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:NFFT
driver.applications.k50Spurious.sense.listPy.range.nfft.set(loffset = 1.0,
↳ rangePy = repcap.RangePy.Default)
```

Defines the number of FFT averages to be performed for each range or segment.

param loffset

integer Range: 1 to 20, Unit: DB

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.18 Pexcursion

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:PEXCursion
```

class PexcursionCls

Pexcursion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:PEXCursion
value: float = driver.applications.k50Spurious.sense.listPy.range.pexcursion.
↳ get(rangePy = repcap.RangePy.Default)
```

Defines the minimum level value by which the signal must rise or fall after a detected spur so that a new spur is detected.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

loffset: Unit: DB

set(loffset: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:PEXCursion
driver.applications.k50Spurious.sense.listPy.range.pexcursion.set(loffset = 1.0,
↪ rangePy = repcap.RangePy.Default)
```

Defines the minimum level value by which the signal must rise or fall after a detected spur so that a new spur is detected.

param loffset

Unit: DB

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.19 RefLevel

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → int

```
# SCPI: [SENSe]:LIST:RANGe<ri>:RLEVel
value: int = driver.applications.k50Spurious.sense.listPy.range.refLevel.
↪ get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

ref_level: Range: -130 dBm to 30 dBm (−10 dBm + RF attenuation – RF preamplifier gain), Unit: DBM

set(ref_level: int, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:RLEVel
driver.applications.k50Spurious.sense.listPy.range.refLevel.set(ref_level = 1,
↪ rangePy = repcap.RangePy.Default)
```

No command help available

param ref_level

Range: -130 dBm to 30 dBm (–10 dBm + RF attenuation – RF preamplifier gain) ,

Unit: DBM

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

6.1.4.12.7.20 SnRatio

SCPI Commands

SENSe:LIST:RANGe<RangePy>:SNRatio

class SnRatioCls

SnRatio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

<pre># SCPI: [SENSe]:LIST:RANGe<ri>:SNRatio value: float = driver.applications.k50Spurious.sense.listPy.range.snRatio. ↳get(rangePy = repcap.RangePy.Default)</pre>

Defines the minimum signal-to-noise ratio (in dB) that the power level must exceed for a spur to be recognized during the final spur frequency scan (see ‘Measurement process’).

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

return

ratio: Unit: DB

set(ratio: float, rangePy=RangePy.Default) → None

<pre># SCPI: [SENSe]:LIST:RANGe<ri>:SNRatio driver.applications.k50Spurious.sense.listPy.range.snRatio.set(ratio = 1.0,↳ ↳rangePy = repcap.RangePy.Default)</pre>

Defines the minimum signal-to-noise ratio (in dB) that the power level must exceed for a spur to be recognized during the final spur frequency scan (see ‘Measurement process’).

param ratio

Unit: DB

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

6.1.4.12.7.21 Threshold

class ThresholdCls

Threshold commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.listPy.range.threshold.clone()
```

Subgroups

6.1.4.12.7.22 Start

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:THReshold:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:THReshold:STARt
value: float = driver.applications.k50Spurious.sense.listPy.range.threshold.
↳start.get(rangePy = repcap.RangePy.Default)
```

Defines an absolute threshold that the power level must exceed for a peak to be detected as a true spur. The start value must be lower than the stop value.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

start: Range: -200 dBm to 0 dBm , Unit: DBM

set(start: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:THReshold:STARt
driver.applications.k50Spurious.sense.listPy.range.threshold.start.set(start =
↳1.0, rangePy = repcap.RangePy.Default)
```

Defines an absolute threshold that the power level must exceed for a peak to be detected as a true spur. The start value must be lower than the stop value.

param start

Range: -200 dBm to 0 dBm , Unit: DBM

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.23 Stop

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:THReshold:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:THReshold:STOP
value: float = driver.applications.k50Spurious.sense.listPy.range.threshold.
↪stop.get(rangePy = repcap.RangePy.Default)
```

Defines an absolute threshold that the power level must exceed for a peak to be detected as a true spur. The stop value must be higher than the start value.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

stop: Range: -200 dBm to 0 dBm , Unit: DBM

set(stop: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:THReshold:STOP
driver.applications.k50Spurious.sense.listPy.range.threshold.stop.set(stop = 1.
↪0, rangePy = repcap.RangePy.Default)
```

Defines an absolute threshold that the power level must exceed for a peak to be detected as a true spur. The stop value must be higher than the start value.

param stop

Range: -200 dBm to 0 dBm , Unit: DBM

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.1.4.12.7.24 UaRange

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:UaRange
```

class UaRangeCls

UaRange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(param: RangeParam, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:UaRange
driver.applications.k50Spurious.sense.listPy.range.uaRange.set(param = enums.
↪RangeParam.ARBW, rangePy = repcap.RangePy.Default)
```

Writes the value of the specified parameter to all of the currently defined ranges.

param param

ARBW | LOFFset | MFRBw | NFFT | PAValue | PEXCursion | RBW | RFATtenuation
| RLEVel | SNRatio | TSTR | TSTP

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface
'Range')

6.1.4.12.8 Measure

class MeasureCls

Measure commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.measure.clone()
```

Subgroups

6.1.4.12.8.1 Points

SCPI Commands

```
SENSe:MEASure:POINts
```

class PointsCls

Points commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MEASure:POINts
value: float = driver.applications.k50Spurious.sense.measure.points.get()
```

Defines the maximum number of trace points within a trace.

return

measurement_points: integer Range: 101 to 32001

set(measurement_points: float) → None

```
# SCPI: [SENSe]:MEASure:POINts
driver.applications.k50Spurious.sense.measure.points.set(measurement_points = 1.
↪0)
```

Defines the maximum number of trace points within a trace.

param measurement_points

integer Range: 101 to 32001

6.1.4.12.9 Mixer

class MixerCls

Mixer commands group definition. 22 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.clone()
```

Subgroups

6.1.4.12.9.1 Bias

class BiasCls

Bias commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.bias.clone()
```

Subgroups

6.1.4.12.9.2 High

SCPI Commands

```
SENSe:MIXer:BIAS:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:BIAS:HIGH
value: float = driver.applications.k50Spurious.sense.mixer.bias.high.get()
```

This command defines the bias current for the high (last) range. . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]).

return

bias_high: No help available

set(bias_high: float) → None

```
# SCPI: [SENSe]:MIXer:BIAS:HIGH
driver.applications.k50Spurious.sense.mixer.bias.high.set(bias_high = 1.0)
```

This command defines the bias current for the high (last) range. . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param bias_high
No help available

6.1.4.12.9.3 Low

SCPI Commands

```
SENSe:MIXer:BIAS:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
value: float = driver.applications.k50Spurious.sense.mixer.bias.low.get()
```

This command defines the bias current for the low (first) range. . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

return
bias_low: No help available

set(bias_low: float) → None

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
driver.applications.k50Spurious.sense.mixer.bias.low.set(bias_low = 1.0)
```

This command defines the bias current for the low (first) range. . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param bias_low
No help available

6.1.4.12.9.4 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.frequency.clone()
```

Subgroups

6.1.4.12.9.5 Handover

SCPI Commands

SENSe:MIXer:FREQuency:HANdOver

class HandoverCls

Handover commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:HANdOver
value: float = driver.applications.k50Spurious.sense.mixer.frequency.handover.
↳get()
```

This command defines the frequency at which the mixer switches from one range to the next (if two different ranges are selected) . The handover frequency for each band can be selected freely within the overlapping frequency range. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

return

handover: No help available

set(handover: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:HANdOver
driver.applications.k50Spurious.sense.mixer.frequency.handover.set(handover = 1.
↳0)
```

This command defines the frequency at which the mixer switches from one range to the next (if two different ranges are selected) . The handover frequency for each band can be selected freely within the overlapping frequency range. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param handover

No help available

6.1.4.12.9.6 Start

SCPI Commands

SENSe:MIXer:FREQuency:STARt

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:STARt
value: float = driver.applications.k50Spurious.sense.mixer.frequency.start.get()
```

This command sets or queries the frequency at which the external mixer band starts.

return
frequency: No help available

set(frequency: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:START
driver.applications.k50Spurious.sense.mixer.frequency.start.set(frequency = 1.0)
```

This command sets or queries the frequency at which the external mixer band starts.

param frequency
No help available

6.1.4.12.9.7 Stop

SCPI Commands

```
SENSe:MIXer:FREQuency:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:STOP
value: float = driver.applications.k50Spurious.sense.mixer.frequency.stop.get()
```

This command sets or queries the frequency at which the external mixer band stops.

return
frequency: No help available

set(frequency: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:STOP
driver.applications.k50Spurious.sense.mixer.frequency.stop.set(frequency = 1.0)
```

This command sets or queries the frequency at which the external mixer band stops.

param frequency
No help available

6.1.4.12.9.8 Harmonic

class HarmonicCls

Harmonic commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.harmonic.clone()
```

Subgroups

6.1.4.12.9.9 Band

SCPI Commands

```
SENSe:MIXer:HARMonic:BAND
SENSe:MIXer:HARMonic:BAND:PRESet
```

class BandCls

Band commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get() → BandB

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
value: enums.BandB = driver.applications.k50Spurious.sense.mixer.harmonic.band.
↳get()
```

This command selects the external mixer band. The query returns the currently selected band. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]).

return

harmonic: No help available

preset() → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.applications.k50Spurious.sense.mixer.harmonic.band.preset()
```

This command restores the preset frequency ranges for the selected standard waveguide band. Note:Changes to the band and mixer settings are maintained even after using the [PRESET] function. Use this command to restore the predefined band ranges.

preset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.applications.k50Spurious.sense.mixer.harmonic.band.preset_with_opc()
```

This command restores the preset frequency ranges for the selected standard waveguide band. Note:Changes to the band and mixer settings are maintained even after using the [PRESET] function. Use this command to restore the predefined band ranges.

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

set(*harmonic: BandB*) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
driver.applications.k50Spurious.sense.mixer.harmonic.band.set(harmonic = enums.
↪BandB.D)
```

This command selects the external mixer band. The query returns the currently selected band. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param harmonic
No help available

6.1.4.12.9.10 High

SCPI Commands

```
SENSe:MIXer:HARMonic:HIGH
```

class HighCls

High commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH
value: float = driver.applications.k50Spurious.sense.mixer.harmonic.high.get()
```

No command help available

return
freq_high: No help available

set(*freq_high: float*) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH
driver.applications.k50Spurious.sense.mixer.harmonic.high.set(freq_high = 1.0)
```

No command help available

param freq_high
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.harmonic.high.clone()
```

Subgroups

6.1.4.12.9.11 State

SCPI Commands

SENSe:MIXer:HARMonic:HIGH:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
value: bool = driver.applications.k50Spurious.sense.mixer.harmonic.high.state.
↳get()
```

This command specifies whether a second (high) harmonic is to be used to cover the band's frequency range.

return

freq_high: No help available

set(freq_high: bool) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
driver.applications.k50Spurious.sense.mixer.harmonic.high.state.set(freq_high =
↳False)
```

This command specifies whether a second (high) harmonic is to be used to cover the band's frequency range.

param freq_high

No help available

6.1.4.12.9.12 Value

SCPI Commands

SENSe:MIXer:HARMonic:HIGH:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH[:VALue]
value: float = driver.applications.k50Spurious.sense.mixer.harmonic.high.value.
↳get()
```

This command specifies the harmonic order to be used for the high (second) range.

return

freq_high: No help available

set(freq_high: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH[:VALue]
driver.applications.k50Spurious.sense.mixer.harmonic.high.value.set(freq_high = 1.0)
```

This command specifies the harmonic order to be used for the high (second) range.

param freq_high
No help available

6.1.4.12.9.13 Low

SCPI Commands

```
SENSe:MIXer:HARMonic:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
value: float = driver.applications.k50Spurious.sense.mixer.harmonic.low.get()
```

This command specifies the harmonic order to be used for the low (first) range.

return
freq_low: No help available

set(freq_low: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
driver.applications.k50Spurious.sense.mixer.harmonic.low.set(freq_low = 1.0)
```

This command specifies the harmonic order to be used for the low (first) range.

param freq_low
No help available

6.1.4.12.9.14 TypePy

SCPI Commands

```
SENSe:MIXer:HARMonic:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → OddEven

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
value: enums.OddEven = driver.applications.k50Spurious.sense.mixer.harmonic.typePy.get()
```

This command specifies whether the harmonic order to be used should be odd, even, or both. Which harmonics are supported depends on the mixer type.

return
type_py: No help available

set(type_py: *OddEven*) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
driver.applications.k50Spurious.sense.mixer.harmonic.typePy.set(type_py = enums.
↪ OddEven.EODD)
```

This command specifies whether the harmonic order to be used should be odd, even, or both. Which harmonics are supported depends on the mixer type.

param type_py
No help available

6.1.4.12.9.15 LoPower

SCPI Commands

SENSe:MIXer:LOPower

class LoPowerCls

LoPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOPower
value: float = driver.applications.k50Spurious.sense.mixer.loPower.get()
```

This command specifies the LO level of the external mixer's LO port.

return
low_power: No help available

set(low_power: *float*) → None

```
# SCPI: [SENSe]:MIXer:LOPower
driver.applications.k50Spurious.sense.mixer.loPower.set(low_power = 1.0)
```

This command specifies the LO level of the external mixer's LO port.

param low_power
No help available

6.1.4.12.9.16 Loss

class LossCls

Loss commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.loss.clone()
```

Subgroups

6.1.4.12.9.17 High

SCPI Commands

```
SENSe:MIXer:LOSS:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
value: float = driver.applications.k50Spurious.sense.mixer.loss.high.get()
```

This command defines the average conversion loss to be used for the entire high (second) range.

return

loss_high: No help available

set(loss_high: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
driver.applications.k50Spurious.sense.mixer.loss.high.set(loss_high = 1.0)
```

This command defines the average conversion loss to be used for the entire high (second) range.

param loss_high

No help available

6.1.4.12.9.18 Low

SCPI Commands

```
SENSe:MIXer:LOSS:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
value: float = driver.applications.k50Spurious.sense.mixer.loss.low.get()
```

This command defines the average conversion loss to be used for the entire low (first) range.

return
loss_low: No help available

set(loss_low: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
driver.applications.k50Spurious.sense.mixer.loss.low.set(loss_low = 1.0)
```

This command defines the average conversion loss to be used for the entire low (first) range.

param loss_low
No help available

6.1.4.12.9.19 Table

class TableCls

Table commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.loss.table.clone()
```

Subgroups

6.1.4.12.9.20 High

SCPI Commands

```
SENSe:MIXer:LOSS:TABLE:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(table_high: str) → str

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE:HIGH
value: str = driver.applications.k50Spurious.sense.mixer.loss.table.high.
    get(table_high = '1')
```

This command defines the conversion loss table to be used for the high (second) range.

param table_high
No help available

return
table_high: No help available

set(table_high: str) → None

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE:HIGH
driver.applications.k50Spurious.sense.mixer.loss.table.high.set(table_high = '1
↪')
```

This command defines the conversion loss table to be used for the high (second) range.

param table_high
No help available

6.1.4.12.9.21 Low

SCPI Commands

```
SENSe:MIXer:LOSS:TABLE:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(table_low: str) → str

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
value: str = driver.applications.k50Spurious.sense.mixer.loss.table.low.
↪get(table_low = '1')
```

This command defines the file name of the conversion loss table to be used for the low (first) range.

param table_low
No help available

return
table_low: No help available

set(table_low: str) → None

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
driver.applications.k50Spurious.sense.mixer.loss.table.low.set(table_low = '1')
```

This command defines the file name of the conversion loss table to be used for the low (first) range.

param table_low
No help available

6.1.4.12.9.22 Ports

SCPI Commands

SENSe:MIXer:PORTs

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:PORTs
value: float = driver.applications.k50Spurious.sense.mixer.ports.get()
```

This command selects the mixer type.

return

port: No help available

set(port: float) → None

```
# SCPI: [SENSe]:MIXer:PORTs
driver.applications.k50Spurious.sense.mixer.ports.set(port = 1.0)
```

This command selects the mixer type.

param port

No help available

6.1.4.12.9.23 RfOvrrange

class RfOvrrangeCls

RfOvrrange commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.mixer.rfOvrrange.clone()
```

Subgroups

6.1.4.12.9.24 State

SCPI Commands

SENSe:MIXer:RFOvrrange:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:RFOVerrange[:STATe]
value: bool = driver.applications.k50Spurious.sense.mixer.rf0verrange.state.
↳ get()
```

If enabled, the band limits are extended beyond ‘RF Start’ and ‘RF Stop’ due to the capabilities of the used harmonics.

return
rf_overrange_state: No help available

set(rf_overrange_state: bool) → None

```
# SCPI: [SENSe]:MIXer:RFOVerrange[:STATe]
driver.applications.k50Spurious.sense.mixer.rf0verrange.state.set(rf_overrange_
↳ state = False)
```

If enabled, the band limits are extended beyond ‘RF Start’ and ‘RF Stop’ due to the capabilities of the used harmonics.

param rf_overrange_state
No help available

6.1.4.12.9.25 Signal

SCPI Commands

SENSe:MIXer:SIGNa1

class SignalCls

Signal commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → State

```
# SCPI: [SENSe]:MIXer:SIGNa1
value: enums.State = driver.applications.k50Spurious.sense.mixer.signal.get()
```

No command help available

return
bias: No help available

set(bias: State) → None

```
# SCPI: [SENSe]:MIXer:SIGNa1
driver.applications.k50Spurious.sense.mixer.signal.set(bias = enums.State.ALL)
```

No command help available

param bias
No help available

6.1.4.12.9.26 State

SCPI Commands

SENSe:MIXer:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer[:STATe]
value: bool = driver.applications.k50Spurious.sense.mixer.state.get()
```

Activates or deactivates the use of a connected external mixer as input for the measurement. This command is only available if the optional External Mixer is installed and an external mixer is connected.

return
state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: [SENSe]:MIXer[:STATe]
driver.applications.k50Spurious.sense.mixer.state.set(state = False)
```

Activates or deactivates the use of a connected external mixer as input for the measurement. This command is only available if the optional External Mixer is installed and an external mixer is connected.

param state
ON | OFF | 1 | 0

6.1.4.12.9.27 Threshold

SCPI Commands

SENSe:MIXer:THReshold

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:THReshold
value: float = driver.applications.k50Spurious.sense.mixer.threshold.get()
```

No command help available

return
threshold: No help available

set(threshold: float) → None

```
# SCPI: [SENSe]:MIXer:THReshold
driver.applications.k50Spurious.sense.mixer.threshold.set(threshold = 1.0)
```

No command help available

param threshold

No help available

6.1.4.12.10 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.sense.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.sense.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 13 total commands, 8 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.clone()
```

Subgroups

6.1.4.12.10.1 Dcycle

class DcycleCls

Dcycle commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.dcycle.clone()
```

Subgroups

6.1.4.12.10.2 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:DCYClE:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.dcycle.state.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

This command turns the duty cycle correction on and off.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(*arg_0: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe[:STATe]
driver.applications.k50Spurious.sense.pmeter.dcycle.state.set(arg_0 = False,
↳ powerMeter = repcap.PowerMeter.Default)
```

This command turns the duty cycle correction on and off.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.3 Value

SCPI Commands

```
SENSe:PMETer<PowerMeter>:DCYCLe:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe:VALue
value: float = driver.applications.k50Spurious.sense.pmeter.dcycle.value.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

This command defines the duty cycle for the correction of pulse signals. The power sensor uses the duty cycle in combination with the mean power to calculate the power of the pulse.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:DCYCLE:VALue
driver.applications.k50Spurious.sense.pmeter.dcycle.value.set(arg_0 = 1.0,
↳powerMeter = repcap.PowerMeter.Default)
```

This command defines the duty cycle for the correction of pulse signals. The power sensor uses the duty cycle in combination with the mean power to calculate the power of the pulse.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.4 Frequency

SCPI Commands

SENSe:PMETer<PowerMeter>:FREQUENCY

class FrequencyCls

Frequency commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → float

```
# SCPI: [SENSe]:PMETer<p>:FREQUENCY
value: float = driver.applications.k50Spurious.sense.pmeter.frequency.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command defines the frequency of the power sensor.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:FREQUENCY
driver.applications.k50Spurious.sense.pmeter.frequency.set(arg_0 = 1.0,
↳powerMeter = repcap.PowerMeter.Default)
```

This command defines the frequency of the power sensor.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.frequency.clone()
```

Subgroups

6.1.4.12.10.5 Link

SCPI Commands

```
SENSe:PMETer<PowerMeter>:FREQuency:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → PmeterFreqLink

```
# SCPI: [SENSe]:PMETer<p>:FREQuency:LINK
value: enums.PmeterFreqLink = driver.applications.k50Spurious.sense.pmeter.
↳ frequency.link.get(powerMeter = repcap.PowerMeter.Default)
```

This command selects the frequency coupling for power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: PmeterFreqLink, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:FREQuency:LINK
driver.applications.k50Spurious.sense.pmeter.frequency.link.set(arg_0 = enums.
↳ PmeterFreqLink.CENTer, powerMeter = repcap.PowerMeter.Default)
```

This command selects the frequency coupling for power sensor measurements.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.6 Mtime

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME
```

class MtimeCls

Mtime commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → Duration

```
# SCPI: [SENSe]:PMETer<p>:MTIME
value: enums.Duration = driver.applications.k50Spurious.sense.pmeter.mtime.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command selects the duration of power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: Duration, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME
driver.applications.k50Spurious.sense.pmeter.mtime.set(arg_0 = enums.Duration.
↳LONG, powerMeter = repcap.PowerMeter.Default)
```

This command selects the duration of power sensor measurements.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.mtime.clone()
```

Subgroups

6.1.4.12.10.7 Average

class AverageCls

Average commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.mtime.average.clone()
```

Subgroups

6.1.4.12.10.8 Count

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME:AVERage:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → float

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage:COUNT
value: float = driver.applications.k50Spurious.sense.pmeter.mtime.average.count.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

This command sets the number of power readings included in the averaging process of power sensor measurements. Extended averaging yields more stable results for power sensor measurements, especially for measurements on signals with a low power, because it minimizes the effects of noise.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage:COUNT
driver.applications.k50Spurious.sense.pmeter.mtime.average.count.set(arg_0 = 1.
↳ 0, powerMeter = repcap.PowerMeter.Default)
```

This command sets the number of power readings included in the averaging process of power sensor measurements. Extended averaging yields more stable results for power sensor measurements, especially for measurements on signals with a low power, because it minimizes the effects of noise.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.9 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME:AVERage:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.mtime.average.state.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command turns averaging for power sensor measurements on and off.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(*arg_0: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage[:STATe]
driver.applications.k50Spurious.sense.pmeter.mtime.average.state.set(arg_0 =
↳False, powerMeter = repcap.PowerMeter.Default)
```

This command turns averaging for power sensor measurements on and off.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.10 Roffset

class RoffsetCls

Roffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.roffset.clone()
```

Subgroups

6.1.4.12.10.11 State

SCPI Commands

SENSe:PMETer<PowerMeter>:ROFFset:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:ROFFset[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.roffset.state.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command includes or excludes the reference level offset of the analyzer for power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(*arg_0: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:ROFFset[:STATe]
driver.applications.k50Spurious.sense.pmeter.roffset.state.set(arg_0 = False,
↳powerMeter = repcap.PowerMeter.Default)
```

This command includes or excludes the reference level offset of the analyzer for power sensor measurements.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.12 Soffset

SCPI Commands

SENSe:PMETer<PowerMeter>:SOFFset

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:SOFFset
value: float = driver.applications.k50Spurious.sense.pmeter.woffset.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

Takes the specified offset into account for the measured power. Only available if [SENSe]:PMETer{p}:ROFFset[:STATe] is disabled.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

sensor_offset: Unit: DB

set(*sensor_offset: float, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:SOFFset
driver.applications.k50Spurious.sense.pmeter.woffset.set(sensor_offset = 1.0,
↳ powerMeter = repcap.PowerMeter.Default)
```

Takes the specified offset into account for the measured power. Only available if [SENSe]:PMETer{p}:ROFFset[:STATe] is disabled.

param sensor_offset

Unit: DB

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.13 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.state.get(powerMeter,
↳ repcap.PowerMeter.Default)
```

This command turns a power sensor on and off.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: bool, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETER<p>[:STATe]
driver.applications.k50Spurious.sense.pmeter.state.set(arg_0 = False,
↳ powerMeter = repcap.PowerMeter.Default)
```

This command turns a power sensor on and off.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.14 Trigger

class TriggerCls

Trigger commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.trigger.clone()
```

Subgroups

6.1.4.12.10.15 Level

SCPI Commands

```
SENSe:PMETER<PowerMeter>:TRIGger:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → float

```
# SCPI: [SENSe]:PMETER<p>:TRIGger:LEVel
value: float = driver.applications.k50Spurious.sense.pmeter.trigger.level.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

This command defines the trigger level for external power triggers. This command requires the use of a Rohde & Schwarz power sensor. For a list of supported sensors, see the datasheet.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:LEVel
driver.applications.k50Spurious.sense.pmeter.trigger.level.set(arg_0 = 1.0,
↳ powerMeter = repcap.PowerMeter.Default)
```

This command defines the trigger level for external power triggers. This command requires the use of a Rohde & Schwarz power sensor. For a list of supported sensors, see the datasheet.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.16 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:TRIGger:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → bool

```
# SCPI: [SENSe]:PMETer<p>:TRIGger[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.trigger.state.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

This command turns the external power trigger on and off. This command requires the use of a Rohde & Schwarz power sensor. For a list of supported sensors, see the data sheet.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: bool, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger[:STATe]
driver.applications.k50Spurious.sense.pmeter.trigger.state.set(arg_0 = False,
↳ powerMeter = repcap.PowerMeter.Default)
```

This command turns the external power trigger on and off. This command requires the use of a Rohde & Schwarz power sensor. For a list of supported sensors, see the data sheet.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.10.17 Update

class UpdateCls

Update commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.pmeter.update.clone()
```

Subgroups

6.1.4.12.10.18 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:UPDate:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → bool

```
# SCPI: [SENSe]:PMETer<p>:UPDate[:STATe]
value: bool = driver.applications.k50Spurious.sense.pmeter.update.state.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command turns continuous update of power sensor measurements on and off. If on, the results are updated even if a single sweep is complete.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: bool, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:UPDate[:STATe]
driver.applications.k50Spurious.sense.pmeter.update.state.set(arg_0 = False,
↳powerMeter = repcap.PowerMeter.Default)
```

This command turns continuous update of power sensor measurements on and off. If on, the results are updated even if a single sweep is complete.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.4.12.11 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k50Spurious.sense.probe.repcap_probe_get()
driver.applications.k50Spurious.sense.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 12 total commands, 2 Subgroups, 0 group commands Repeated Capability: Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.probe.clone()
```

Subgroups

6.1.4.12.11.1 Id

class IdCls

Id commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.probe.id.clone()
```

Subgroups

6.1.4.12.11.2 PartNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:PARTnumber
```

class PartNumberCls

PartNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:ID:PARTnumber
value: float = driver.applications.k50Spurious.sense.probe.id.partNumber.
    ↪ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

part_number: No help available

6.1.4.12.11.3 SrNumber

SCPI Commands

SENSe:PROBe<Probe>:ID:SRNumber

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → str

```
# SCPI: [SENSe]:PROBe<pb>:ID:SRNumber
value: str = driver.applications.k50Spurious.sense.probe.id.srNumber.get(probe_
↳ = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

serial_no: No help available

6.1.4.12.11.4 Setup

class SetupCls

Setup commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.probe.setup.clone()
```

Subgroups

6.1.4.12.11.5 AttRatio

SCPI Commands

SENSe:PROBe<Probe>:SETup:ATTRatio

class AttRatioCls

AttRatio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
value: float = driver.applications.k50Spurious.sense.probe.setup.attRatio.
↪get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

attenuation_ratio: No help available

set(*attenuation_ratio: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
driver.applications.k50Spurious.sense.probe.setup.attRatio.set(attenuation_
↪ratio = 1.0, probe = repcap.Probe.Default)
```

No command help available

param attenuation_ratio

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.6 CmOffset**SCPI Commands**

```
SENSe:PROBe<Probe>:SETup:CMOffset
```

class CmOffsetCls

CmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
value: float = driver.applications.k50Spurious.sense.probe.setup.cmOffset.
↪get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

cm_offset: No help available

set(*cm_offset*: float, *probe*=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
driver.applications.k50Spurious.sense.probe.setup.cmOffset.set(cm_offset = 1.0,
↪probe = repcap.Probe.Default)
```

No command help available

param cm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.7 DmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:DMOffset
```

class DmOffsetCls

DmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe*=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOffset
value: float = driver.applications.k50Spurious.sense.probe.setup.dmOffset.
↪get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

dm_offset: No help available

set(*dm_offset*: float, *probe*=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOffset
driver.applications.k50Spurious.sense.probe.setup.dmOffset.set(dm_offset = 1.0,
↪probe = repcap.Probe.Default)
```

No command help available

param dm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.8 Mode

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeSetupMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
value: enums.ProbeSetupMode = driver.applications.k50Spurious.sense.probe.setup.
↳mode.get(probe = reproc.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(*mode: ProbeSetupMode, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
driver.applications.k50Spurious.sense.probe.setup.mode.set(mode = enums.
↳ProbeSetupMode.NOAction, probe = reproc.Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.9 Name

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → str

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NAME
value: str = driver.applications.k50Spurious.sense.probe.setup.name.get(probe =
↳reproc.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

name: No help available

6.1.4.12.11.10 NmOffset

SCPI Commands

`SENSe:PROBe<Probe>:SETup:NMOffset`

class NmOffsetCls

NmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset
value: float = driver.applications.k50Spurious.sense.probe.setup.nmOffset.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

nm_offset: No help available

set(*nm_offset: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset
driver.applications.k50Spurious.sense.probe.setup.nmOffset.set(nm_offset = 1.0,
↳ probe = repcap.Probe.Default)
```

No command help available

param nm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.11 Pmode

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMODE
```

class PmodeCls

Pmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
value: enums.ProbeMode = driver.applications.k50Spurious.sense.probe.setup.
↳ pmode.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(*mode: ProbeMode, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
driver.applications.k50Spurious.sense.probe.setup.pmode.set(mode = enums.
↳ ProbeMode.CM, probe = repcap.Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.12 PmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMOffset
```

class PmOffsetCls

PmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
value: float = driver.applications.k50Spurious.sense.probe.setup.pmOffset.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

pm_offset: No help available

set(*pm_offset*: float, *probe*=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
driver.applications.k50Spurious.sense.probe.setup.pmOffset.set(pm_offset = 1.0,
↪ probe = repcap.Probe.Default)
```

No command help available

param pm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.4.12.11.13 State

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe*=Probe.Default) → Detect

```
# SCPI: [SENSe]:PROBe<pb>:SETup:STATe
value: enums.Detect = driver.applications.k50Spurious.sense.probe.setup.state.
↪ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

state: No help available

6.1.4.12.11.14 TypePy

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → str

```
# SCPI: [SENSe]:PROBe<pb>:SETup:TYPE
value: str = driver.applications.k50Spurious.sense.probe.setup.typePy.get(probe_
↳ repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

type_py: No help available

6.1.4.12.12 Ssearch

class SsearchCls

Ssearch commands group definition. 7 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.ssearch.clone()
```

Subgroups

6.1.4.12.12.1 Control

SCPI Commands

```
SENSe:SSEarch:CONTRol
```

class ControlCls

Control commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → MeasurementStep

```
# SCPI: [SENSe]:SSEarch:CONTRol
value: enums.MeasurementStep = driver.applications.k50Spurious.sense.ssearch.
↳ control.get()
```

Defines which steps of the measurement process are performed. All steps up to the selected step are performed. By default, all measurement steps are performed. For details on the measurement process steps see ‘Measurement process’.

return

step: SOVerview | NESTimate | SDETection | SPOTstep SOVerview Spectral overview only NESTimate Spectral overview and Noise Floor Estimation SDETection Spectral overview, Noise Floor Estimation, and Spurious Detection measurement SPOT Spot Search - all measurement steps are performed

set(step: MeasurementStep) → None

```
# SCPI: [SENSe]:SSEarch:CONTRol
driver.applications.k50Spurious.sense.ssearch.control.set(step = enums.
↳ MeasurementStep.NESTimate)
```

Defines which steps of the measurement process are performed. All steps up to the selected step are performed. By default, all measurement steps are performed. For details on the measurement process steps see ‘Measurement process’.

param step

SOVerview | NESTimate | SDETection | SPOTstep SOVerview Spectral overview only NESTimate Spectral overview and Noise Floor Estimation SDETection Spectral overview, Noise Floor Estimation, and Spurious Detection measurement SPOT Spot Search - all measurement steps are performed

6.1.4.12.12.2 Fplan

SCPI Commands

SENSe:SSEarch:FPLan

class FplanCls

Fplan commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SSEarch:FPLan
value: bool = driver.applications.k50Spurious.sense.ssearch.fplan.get()
```

Enables or disables the the use of the frequency plan for identification of spurs.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:SSEarch:FPLan
driver.applications.k50Spurious.sense.ssearch.fplan.set(state = False)
```

Enables or disables the the use of the frequency plan for identification of spurs.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.ssearch.fplan.clone()
```

Subgroups

6.1.4.12.12.3 Tolerance

SCPI Commands

```
SENSe:SSEarch:FPLan:TOLerance
```

class ToleranceCls

Tolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SSEarch:FPLan:TOLerance
value: float = driver.applications.k50Spurious.sense.ssearch.fplan.tolerance.
↳ get()
```

Sets the frequency tolerance to match predicted spurs to measured spurs.

return

frequency: numeric value Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:SSEarch:FPLan:TOLerance
driver.applications.k50Spurious.sense.ssearch.fplan.tolerance.set(frequency = 1.
↳ 0)
```

Sets the frequency tolerance to match predicted spurs to measured spurs.

param frequency

numeric value Unit: Hz

6.1.4.12.12.4 Mspur

SCPI Commands

```
SENSe:SSEarch:MSPur
```

class MspurCls

Mspur commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → MspurSearchType

```
# SCPI: [SENSe]:SSEarch:MSPur
value: enums.MspurSearchType = driver.applications.k50Spurious.sense.ssearch.
↳ mspur.get()
```

Defines the condition for matching the measured to the predicted spurs.

return

type_py: DMINimum | PMAximum DMINimum If multiple measured spurs are inside the tolerance range around a predicted spur, the measured spur closest to the predicted spur is identified as the predicted. PMAximum If multiple measured spurs are inside the tolerance range around a predicted spur, the measured spur with the highest power will be identified as the predicted.

set(type_py: MspurSearchType) → None

```
# SCPI: [SENSe]:SSEarch:MSPur
driver.applications.k50Spurious.sense.ssearch.mspur.set(type_py = enums.
↳ MspurSearchType.DMINimum)
```

Defines the condition for matching the measured to the predicted spurs.

param type_py

DMINimum | PMAximum DMINimum If multiple measured spurs are inside the tolerance range around a predicted spur, the measured spur closest to the predicted spur is identified as the predicted. PMAximum If multiple measured spurs are inside the tolerance range around a predicted spur, the measured spur with the highest power will be identified as the predicted.

6.1.4.12.12.5 Rmark

SCPI Commands

```
SENSe:SSEarch:RMark
```

class RmarkCls

Rmark commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SSEarch:RMark
value: bool = driver.applications.k50Spurious.sense.ssearch.rmark.get()
```

No command help available

return

state: ON | OFF | 0 | 1 OFF | 0 Residuals are not marked ON | 1 Residuals are marked

set(state: bool) → None

```
# SCPI: [SENSe]:SSEarch:RMark
driver.applications.k50Spurious.sense.ssearch.rmark.set(state = False)
```

No command help available

param state

ON | OFF | 0 | 1 OFF | 0 Residuals are not marked ON | 1 Residuals are marked

6.1.4.12.12.6 Rremove

SCPI Commands

SENSe:SSEarch:RREMove

class RremoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SSEarch:RREMove
value: bool = driver.applications.k50Spurious.sense.ssearch.rremove.get()
```

If enabled, residual spurs, which are generated by internal components in the R&S FSWP itself, are not included in the spur results. Note, however, if a residual spur coincides with a ‘true’ spur, the spur is also removed.

return

state: ON | OFF | 0 | 1 OFF | 0 Residuals are not removed ON | 1 Residuals are removed

set(state: bool) → None

```
# SCPI: [SENSe]:SSEarch:RREMove
driver.applications.k50Spurious.sense.ssearch.rremove.set(state = False)
```

If enabled, residual spurs, which are generated by internal components in the R&S FSWP itself, are not included in the spur results. Note, however, if a residual spur coincides with a ‘true’ spur, the spur is also removed.

param state

ON | OFF | 0 | 1 OFF | 0 Residuals are not removed ON | 1 Residuals are removed

6.1.4.12.12.7 Stype

SCPI Commands

SENSe:SSEarch:STYPe

class StypeCls

Stype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → MeasurementType

```
# SCPI: [SENSe]:SSEarch:STYPe
value: enums.MeasurementType = driver.applications.k50Spurious.sense.ssearch.
↳stype.get()
```

Defines the type of measurement to be configured and performed.

return

type_py: WIDE | DIRected WIDE A measurement with a large span to detect any possible spurs in the entire frequency span of an input signal. This measurement is useful if you have little or no knowledge of the current input signal or where to expect

spurs, and require an overview. DIRected A measurement performed at predefined discrete frequencies with settings optimized for the current signal and noise levels at those frequencies. This measurement is targeted at determining the precise level and exact frequency of spurs that are basically already known or expected.

set(type_py: *MeasurementType*) → None

```
# SCPI: [SENSe]:SSEarch:STYPe
driver.applications.k50Spurious.sense.ssearch.type.set(type_py = enums.
↳ MeasurementType.DIRected)
```

Defines the type of measurement to be configured and performed.

param type_py

WIDE | DIRected WIDE A measurement with a large span to detect any possible spurs in the entire frequency span of an input signal. This measurement is useful if you have little or no knowledge of the current input signal or where to expect spurs, and require an overview. DIRected A measurement performed at predefined discrete frequencies with settings optimized for the current signal and noise levels at those frequencies. This measurement is targeted at determining the precise level and exact frequency of spurs that are basically already known or expected.

6.1.4.12.13 Transfer

class TransferCls

Transfer commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.sense.transfer.clone()
```

Subgroups

6.1.4.12.13.1 Segment

SCPI Commands

```
SENSe:TRANsfer:SEGment
```

class SegmentCls

Segment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:TRANsfer:SEGment
driver.applications.k50Spurious.sense.transfer.segment.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:TRANsfer:SEGment
driver.applications.k50Spurious.sense.transfer.segment.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.12.13.2 Spur

SCPI Commands

```
SENSe:TRANsfer:SPUR
```

class SpurCls

Spur commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(spur: Optional[List[int]] = None) → None

```
# SCPI: [SENSe]:TRANsfer:SPUR
driver.applications.k50Spurious.sense.transfer.spur.set(spur = [1, 2, 3])
```

No command help available

param spur

Comma-separated list of spur numbers (integers)

6.1.4.13 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.trace.repcap_window_get()
driver.applications.k50Spurious.trace.repcap_window_set(repcap.Window.Nr1)
```

class TraceCls

Trace commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trace.clone()
```

Subgroups

6.1.4.13.1 Data

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA
```

class DataCls

Data commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(trace_type: TraceTypeList, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.applications.k50Spurious.trace.data.get(trace_type,
↪= enums.TraceTypeList.LIST, window = repcap.Window.Default)
```

This command queries the y-values in the selected result display. The unit depends on the display and on the unit you have currently set.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_ydata: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trace.data.clone()
```

Subgroups

6.1.4.13.1.1 X

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace_type*: *TraceTypeNumeric*, *window*=*Window.Default*) → List[float]

```
# SCPI: TRACe<n>[:DATA]:X
value: List[float] = driver.applications.k50Spurious.trace.data.x.get(trace_
↪type = enums.TraceTypeNumeric.TRACe1, window = repcap.Window.Default)
```

This remote control command returns the X values only for the trace in the selected result display. Depending on the type of result display and the scaling of the x-axis, this can be either the pulse number or a timestamp for each detected pulse in the capture buffer. This command is only available for graphical displays, except for the Magnitude Capture display.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_xdata: No help available

6.1.4.14 Trigger

class TriggerCls

Trigger commands group definition. 10 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.clone()
```

Subgroups

6.1.4.14.1 Sequence

class SequenceCls

Sequence commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.sequence.clone()
```

Subgroups

6.1.4.14.1.1 Dtime

SCPI Commands

TRIGger:SEquence:DTIME

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:DTIME
value: float = driver.applications.k50Spurious.trigger.sequence.dtime.get()
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

return

time: No help available

set(time: float) → None

```
# SCPI: TRIGger[:SEquence]:DTIME
driver.applications.k50Spurious.trigger.sequence.dtime.set(time = 1.0)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param time

Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

6.1.4.14.1.2 Holdoff

class HoldoffCls

Holdoff commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.sequence.holdoff.clone()
```

Subgroups

6.1.4.14.1.3 Time

SCPI Commands

TRIGger:SEquence:HOLDoff:TIME

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
value: float = driver.applications.k50Spurious.trigger.sequence.holdoff.time.
↪get()
```

Defines the time offset between the trigger event and the start of the measurement.

return
time: No help available

set(time: float) → None

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
driver.applications.k50Spurious.trigger.sequence.holdoff.time.set(time = 1.0)
```

Defines the time offset between the trigger event and the start of the measurement.

param time
Unit: S

6.1.4.14.1.4 IfPower**class IfPowerCls**

IfPower commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.sequence.ifPower.clone()
```

Subgroups**6.1.4.14.1.5 Holdoff****SCPI Commands**

```
TRIGger:SEquence:IFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
value: float = driver.applications.k50Spurious.trigger.sequence.ifPower.holdoff.
↪get()
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) .

return
time: No help available

set(time: float) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
driver.applications.k50Spurious.trigger.sequence.ifPower.holdoff.set(time = 1.0)
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) .

param time
Range: 0 s to 10 s, Unit: S

6.1.4.14.1.6 Hysteresis

SCPI Commands

```
TRIGger:SEquence:IFPower:HYSTeresis
```

class HysteresisCls

Hysteresis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
value: float = driver.applications.k50Spurious.trigger.sequence.ifPower.
↳hysteresis.get()
```

This command defines the trigger hysteresis, which is only available for ‘IF Power’ trigger sources.

return
hysteresis: Range: 3 dB to 50 dB, Unit: DB

set(hysteresis: float) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
driver.applications.k50Spurious.trigger.sequence.ifPower.hysteresis.
↳set(hysteresis = 1.0)
```

This command defines the trigger hysteresis, which is only available for ‘IF Power’ trigger sources.

param hysteresis
Range: 3 dB to 50 dB, Unit: DB

6.1.4.14.1.7 Level

class LevelCls

Level commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.sequence.level.clone()
```

Subgroups

6.1.4.14.1.8 External<ExternalPort>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.applications.k50Spurious.trigger.sequence.level.external.repcap_externalPort_
↪get()
driver.applications.k50Spurious.trigger.sequence.level.external.repcap_externalPort_
↪set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
TRIGger:SEquence:LEVel:EXTernal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(externalPort=ExternalPort.Default) → float

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXTernal<port>]
value: float = driver.applications.k50Spurious.trigger.sequence.level.external.
↪get(externalPort = repcap.ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'External')

return

level: No help available

set(level: float, externalPort=ExternalPort.Default) → None

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXTernal<port>]
driver.applications.k50Spurious.trigger.sequence.level.external.set(level = 1.0,
↪ externalPort = repcap.ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param level

No help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'External')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.trigger.sequence.level.external.clone()
```

6.1.4.14.1.9 IfPower

SCPI Commands

```
TRIGger:SEquence:LEVel:IFPower
```

class IfPowerCls

IfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
value: float = driver.applications.k50Spurious.trigger.sequence.level.ifPower.
    ↪get()
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

return

level: No help available

set(level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
driver.applications.k50Spurious.trigger.sequence.level.ifPower.set(level = 1.0)
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.1.4.14.1.10 IqPower

SCPI Commands

```
TRIGger:SEquence:LEVel:IQPower
```

class IqPowerCls

IqPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
value: float = driver.applications.k50Spurious.trigger.sequence.level.iqPower.
↳get()
```

No command help available

return
level: No help available

set(level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
driver.applications.k50Spurious.trigger.sequence.level.iqPower.set(level = 1.0)
```

No command help available

param level
No help available

6.1.4.14.1.11 RfPower

SCPI Commands

```
TRIGger:SEquence:LEVel:RFPower
```

class RfPowerCls

RfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:RFPower
value: float = driver.applications.k50Spurious.trigger.sequence.level.rfPower.
↳get()
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

return
level: No help available

set(*level: float*) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:RFPower
driver.applications.k50Spurious.trigger.sequence.level.rfPower.set(level = 1.0)
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

param level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.1.4.14.1.12 Slope

SCPI Commands

```
TRIGger:SEquence:SLOPe
```

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SlopeType

```
# SCPI: TRIGger[:SEquence]:SLOPe
value: enums.SlopeType = driver.applications.k50Spurious.trigger.sequence.slope.
↳get()
```

This command selects the trigger slope.

return

slope: No help available

set(*slope: SlopeType*) → None

```
# SCPI: TRIGger[:SEquence]:SLOPe
driver.applications.k50Spurious.trigger.sequence.slope.set(slope = enums.
↳SlopeType.NEGative)
```

This command selects the trigger slope.

param slope

POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

6.1.4.14.13 Source

SCPI Commands

```
TRIGger:SEquence:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerSourceL

```
# SCPI: TRIGger[:SEquence]:SOURce
value: enums.TriggerSourceL = driver.applications.k50Spurious.trigger.sequence.
↪source.get()
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

return

source: IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’.

set(source: TriggerSourceL) → None

```
# SCPI: TRIGger[:SEquence]:SOURce
driver.applications.k50Spurious.trigger.sequence.source.set(source = enums.
↪TriggerSourceL.EXT2)
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

param source

IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’.

6.1.4.15 TriggerInvoke

SCPI Commands

```
*TRG
```

class TriggerInvokeCls

TriggerInvoke commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: *TRG
driver.applications.k50Spurious.triggerInvoke.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: *TRG
driver.applications.k50Spurious.triggerInvoke.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.4.16 Unit

class UnitCls

Unit commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.unit.clone()
```

Subgroups

6.1.4.16.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k50Spurious.unit.pmeter.repcap_powerMeter_get()
driver.applications.k50Spurious.unit.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.unit.pmeter.clone()
```


Subgroups

6.1.4.16.1.1 Power

SCPI Commands

```
UNIT:PMETer<PowerMeter>:POWer
```

class PowerCls

Power commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → PowerMeterUnit

```
# SCPI: UNIT:PMETer<p>:POWer
value: enums.PowerMeterUnit = driver.applications.k50Spurious.unit.pmeter.power.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command selects the unit for absolute power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(arg_0: PowerMeterUnit, powerMeter=PowerMeter.Default) → None

```
# SCPI: UNIT:PMETer<p>:POWer
driver.applications.k50Spurious.unit.pmeter.power.set(arg_0 = enums.
↳PowerMeterUnit.DBM, powerMeter = repcap.PowerMeter.Default)
```

This command selects the unit for absolute power sensor measurements.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k50Spurious.unit.pmeter.power.clone()
```

Subgroups

6.1.4.16.1.2 Ratio

SCPI Commands

UNIT:PMETer<PowerMeter>:POWer:RATio

class RatioCls

Ratio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → UnitMode

```
# SCPI: UNIT:PMETer<p>:POWer:RATio
value: enums.UnitMode = driver.applications.k50Spurious.unit.pmeter.power.ratio.
↳get(powerMeter = repcap.PowerMeter.Default)
```

This command selects the unit for relative power sensor measurements.

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

arg_0: No help available

set(*arg_0: UnitMode, powerMeter=PowerMeter.Default*) → None

```
# SCPI: UNIT:PMETer<p>:POWer:RATio
driver.applications.k50Spurious.unit.pmeter.power.ratio.set(arg_0 = enums.
↳UnitMode.DB, powerMeter = repcap.PowerMeter.Default)
```

This command selects the unit for relative power sensor measurements.

param arg_0

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.1.5 K60_Transient

class K60_TransientCls

K60_Transient commands group definition. 187 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60_Transient.clone()
```

Subgroups

6.1.5.1 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.calculate.repcap_window_get()
driver.applications.k60Transient.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 74 total commands, 6 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.clone()
```

Subgroups

6.1.5.1.1 ChrDetection

class ChrDetectionCls

ChrDetection commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.chrDetection.clone()
```

Subgroups

6.1.5.1.1.1 Pnoise

class PnoiseCls

Pnoise commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.chrDetection.pnoise.clone()
```

Subgroups

6.1.5.1.1.2 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.chrDetection.pnoise.frequency.clone()
```

Subgroups

6.1.5.1.1.3 Start

SCPI Commands

```
CALCulate<Window>:CHRDetection:PNOise:FREQUENCY:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:FREQUENCY:START
value: float = driver.applications.k60Transient.calculate.chrDetection.pnoise.
    ↪ frequency.start.get(window = repcap.Window.Default)
```

Sets the start frequency for the phase noise chirp measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

frequency: Unit: Hz

set(frequency: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:FREQUENCY:START
driver.applications.k60Transient.calculate.chrDetection.pnoise.frequency.start.
    ↪ set(frequency = 1.0, window = repcap.Window.Default)
```

Sets the start frequency for the phase noise chirp measurement.

param frequency

Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.1.4 Stop**SCPI Commands**

CALCulate<Window>:CHRDetection:PNOise:FREQuency:STOP

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:FREQuency:STOP
value: float = driver.applications.k60Transient.calculate.chrDetection.pnoise.
    frequency.stop.get(window = repcap.Window.Default)
```

Sets the stop frequency for the phase noise chirp measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

frequency: Unit: Hz

set(frequency: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:FREQuency:STOP
driver.applications.k60Transient.calculate.chrDetection.pnoise.frequency.stop.
    set(frequency = 1.0, window = repcap.Window.Default)
```

Sets the stop frequency for the phase noise chirp measurement.

param frequency

Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.1.5 Length

SCPI Commands

`CALCulate<Window>:CHRDetection:PNOise:LENGth`

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:LENGth
value: float = driver.applications.k60Transient.calculate.chrDetection.pnoise.
↳length.get(window = repcap.Window.Default)
```

Defines the length of the measurement range for power results in percent of the chirp length. This command is only available if the reference is CENT (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Reference.set).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

percent: percent of the chirp length Range: 0 to 100

set(*percent: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:LENGth
driver.applications.k60Transient.calculate.chrDetection.pnoise.length.
↳set(percent = 1.0, window = repcap.Window.Default)
```

Defines the length of the measurement range for power results in percent of the chirp length. This command is only available if the reference is CENT (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Reference.set).

param percent

percent of the chirp length Range: 0 to 100

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.1.6 Offset

class OffsetCls

Offset commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.chrDetection.pnoise.offset.clone()
```

Subgroups

6.1.5.1.1.7 Begin

SCPI Commands

```
CALCulate<Window>:CHRDetection:PNOise:OFFSet:BEgin
```

class BeginCls

Begin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:OFFSet:BEgin
value: float = driver.applications.k60Transient.calculate.chrDetection.pnoise.
↳offset.begin.get(window = repcap.Window.Default)
```

Defines the beginning of the measurement range for phase noise results as an offset in seconds from the chirp start. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection. Pnoise.Reference.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

time: Unit: S

set(time: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:OFFSet:BEgin
driver.applications.k60Transient.calculate.chrDetection.pnoise.offset.begin.
↳set(time = 1.0, window = repcap.Window.Default)
```

Defines the beginning of the measurement range for phase noise results as an offset in seconds from the chirp start. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection. Pnoise.Reference.set) .

param time

Unit: S

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.1.8 End

SCPI Commands

`CALCulate<Window>:CHRDetection:PNOise:OFFSet:END`

class EndCls

End commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:OFFSet:END
value: float = driver.applications.k60Transient.calculate.chrDetection.pnoise.
    ↪offset.end.get(window = repcap.Window.Default)
```

Defines the end of the measurement range for phase noise results as an offset in seconds from the chirp end. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Reference.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

time: Unit: S

set(*time: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:OFFSet:END
driver.applications.k60Transient.calculate.chrDetection.pnoise.offset.end.
    ↪set(time = 1.0, window = repcap.Window.Default)
```

Defines the end of the measurement range for phase noise results as an offset in seconds from the chirp end. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Reference.set) .

param time

Unit: S

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.1.9 Reference

SCPI Commands

`CALCulate<Window>:CHRDetection:PNOise:REference`

class ReferenceCls

Reference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → ResultDevReference

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:REference
value: enums.ResultDevReference = driver.applications.k60Transient.calculate.
↳ chrDetection.pnoise.reference.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

reference: CENTER | EDGE | FMSettling | PMSettling
EDGE The measurement range is defined in reference to the chirp’s rising or falling edge (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Offset.Begin.set and method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Offset.End.set)
. CENTER The measurement range is defined in reference to the center of the chirp. FMSettling The measurement range starts at the FM settling point (see [SENSe:]HOP:FMSettling:FMSPoint?) . PMSettling The measurement range starts at the PM settling point (see [SENSe:]HOP:PMSettling:PMSPoint?) .

set(reference: ResultDevReference, window=Window.Default) → None

```
# SCPI: CALCulate<n>:CHRDetection:PNOise:REference
driver.applications.k60Transient.calculate.chrDetection.pnoise.reference.
↳ set(reference = enums.ResultDevReference.CENTER, window = repcap.Window.
↳ Default)
```

No command help available

param reference

CENTER | EDGE | FMSettling | PMSettling
EDGE The measurement range is defined in reference to the chirp’s rising or falling edge (see method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Offset.Begin.set and method RsFswp.Applications.K60_Transient.Calculate.ChrDetection.Pnoise.Offset.End.set)
. CENTER The measurement range is defined in reference to the center of the chirp. FMSettling The measurement range starts at the FM settling point (see [SENSe:]HOP:FMSettling:FMSPoint?) . PMSettling The measurement range starts at the PM settling point (see [SENSe:]HOP:PMSettling:PMSPoint?) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.2 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k60Transient.calculate.deltaMarker.repcap_deltaMarker_get()
driver.applications.k60Transient.calculate.deltaMarker.repcap_deltaMarker_set(repcap.
↳ DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 27 total commands, 9 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.clone()
```

Subgroups**6.1.5.1.2.1 Aoff****SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:AOFF
driver.applications.k60Transient.calculate.deltaMarker.aoff.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns off all delta markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.2 LinkTo**class LinkToCls**

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.linkTo.clone()
```

Subgroups

6.1.5.1.2.3 Marker<MarkerDestination>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.calculate.deltaMarker.linkTo.marker.repcap_
↪markerDestination_get()
driver.applications.k60Transient.calculate.deltaMarker.linkTo.marker.repcap_
↪markerDestination_set(repcap.MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:LINK:TO:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

get(window=Window.Default, deltaMarker=DeltaMarker.Default, markerDestination=MarkerDestination.Default) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m1>:LINK:TO:MARKer<m2>
value: bool = driver.applications.k60Transient.calculate.deltaMarker.linkTo.
↪marker.get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default, markerDestination = repcap.MarkerDestination.Default)
```

This command links the delta source marker <ms> to any active destination marker <md> (normal or delta marker).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m1>:LINK:TO:MARKer<m2>
driver.applications.k60Transient.calculate.deltaMarker.linkTo.marker.set(state,
↪= False, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default, markerDestination = repcap.MarkerDestination.Default)
```

This command links the delta source marker <ms> to any active destination marker <md> (normal or delta marker) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.linkTo.marker.clone()
```

6.1.5.1.2.4 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.maximum.clone()
```

Subgroups

6.1.5.1.2.5 Left

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:LEFT
driver.applications.k60Transient.calculate.deltaMarker.maximum.left.set(window,
↪= repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.6 Next**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:NEXT
driver.applications.k60Transient.calculate.deltaMarker.maximum.next.set(window,
↪= repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next positive peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.7 Peak

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.maximum.peak.set(window_
↳ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.8 Right

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:RIGHT
driver.applications.k60Transient.calculate.deltaMarker.maximum.right.set(window_
↳ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value on the trace. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.9 Minimum**class MinimumCls**

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.minimum.clone()
```

Subgroups**6.1.5.1.2.10 Left****SCPI Commands**

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:LEFT
driver.applications.k60Transient.calculate.deltaMarker.minimum.left.set(window_
↳ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.11 Next

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:NEXT
driver.applications.k60Transient.calculate.deltaMarker.minimum.next.set(window,
↳ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.12 Peak

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.minimum.peak.set(window,
↳ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.13 Right

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:RIGHT
driver.applications.k60Transient.calculate.deltaMarker.minimum.right.set(window=
↳ repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.2.14 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.clone()
```

Subgroups

6.1.5.1.2.15 Frame

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECTrogram:FRAME
```

class FrameCls

Frame commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECTrogram:FRAME
value: float = driver.applications.k60Transient.calculate.deltaMarker.
↪ spectrogram.frame.get(window = repcap.Window.Default, deltaMarker = repcap.
↪ DeltaMarker.Default)
```

This command positions a delta marker on a particular frame. The frame is relative to the position of marker 1. The command is available for the spectrogram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

frame: Selects a frame either by its frame number or time stamp. The frame number is available if the time stamp is off. The range depends on the history depth. The time stamp is available if the time stamp is on. The number is the distance to frame 0 in seconds. The range depends on the history depth. Unit: S

set(frame: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECTrogram:FRAME
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.frame.
↪ set(frame = 1.0, window = repcap.Window.Default, deltaMarker = repcap.
↪ DeltaMarker.Default)
```

This command positions a delta marker on a particular frame. The frame is relative to the position of marker 1. The command is available for the spectrogram.

param frame

Selects a frame either by its frame number or time stamp. The frame number is available if the time stamp is off. The range depends on the history depth. The time stamp is available if the time stamp is on. The number is the distance to frame 0 in seconds. The range depends on the history depth. Unit: S

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.1.5.1.2.16 Sarea

SCPI Commands

CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:SARea

class SareaCls

Sarea commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → SearchArea

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:SARea
value: enums.SearchArea = driver.applications.k60Transient.calculate.
↳ deltaMarker.spectrogram.sarea.get(window = repcap.Window.Default, deltaMarker.
↳ = repcap.DeltaMarker.Default)
```

This command defines the marker search area for all spectrogram markers in the channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

search_area: VISible Performs a search within the visible frames. Note that the command does not work if the spectrogram is not visible for any reason (e.g. if the display update is off) . MEMory Performs a search within all frames in the memory.

set(search_area: SearchArea, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:SARea
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.sarea.
↳ set(search_area = enums.SearchArea.MEMory, window = repcap.Window.Default,
↳ deltaMarker = repcap.DeltaMarker.Default)
```

This command defines the marker search area for all spectrogram markers in the channel.

param search_area

VISible Performs a search within the visible frames. Note that the command does not work if the spectrogram is not visible for any reason (e.g. if the display update is off) . MEMory Performs a search within all frames in the memory.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.5.1.2.17 Xy

class XyCls

Xy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.xy.clone()
```

Subgroups

6.1.5.1.2.18 Maximum

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.xy.maximum.
↳ clone()
```

Subgroups

6.1.5.1.2.19 Peak

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:XY:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:XY:MAXimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.xy.maximum.
↳ peak.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳ Default)
```

This command moves a marker to the highest level of the spectrogram over all frequencies.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.20 Minimum**class MinimumCls**

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.xy.minimum.
↳ clone()
```

Subgroups**6.1.5.1.2.21 Peak****SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:XY:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:XY:MINimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.xy.minimum.
↳ peak.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳ Default)
```

This command moves a delta marker to the minimum level of the spectrogram over all frequencies.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.22 Y

class YCls

Y commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.clone()
```

Subgroups

6.1.5.1.2.23 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.maximum.
↳clone()
```

Subgroups

6.1.5.1.2.24 Above

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:ABOVe
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum:ABOVe
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.maximum.
↳above.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a marker vertically to the next higher level for the current frequency. The search includes only frames above the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.2.25 Below**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum:BELOW
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.maximum.
↳below.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a marker vertically to the next higher level for the current frequency. The search includes only frames below the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.2.26 Next**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum:NEXT
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.maximum.
↳next.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a delta marker vertically to the next higher level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.2.27 Peak

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.maximum.
↳peak.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a delta marker vertically to the highest level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker. If the marker hasn't been active yet, the command looks for the peak level in the whole spectrogram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.2.28 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.minimum.
↳ clone()
```

Subgroups

6.1.5.1.2.29 Above

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:Y:MINimum:ABOVe
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:Y:MINimum:ABOVe
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.minimum.
↳ above.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳ Default)
```

This command moves a delta marker vertically to the next minimum level for the current frequency. The search includes only frames above the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.30 Below

SCPI Commands

```
CALCulate<Window>:DELTAmarker<DeltaMarker>:SPECtrogram:Y:MINimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>:SPECtrogram:Y:MINimum:BELOW
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.minimum.
↳below.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a delta marker vertically to the next minimum level for the current frequency. The search includes only frames below the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.31 Next

SCPI Commands

```
CALCulate<Window>:DELTAmarker<DeltaMarker>:SPECtrogram:Y:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>:SPECtrogram:Y:MINimum:NEXT
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.minimum.
↳next.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

This command moves a delta marker vertically to the next minimum level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.32 Peak**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MINimum[:PEAK]
driver.applications.k60Transient.calculate.deltaMarker.spectrogram.y.minimum.
↪peak.set(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default)
```

This command moves a delta marker vertically to the minimum level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker. If the marker hasn't been active yet, the command first looks for the peak level in the whole spectrogram and moves the marker vertically to the minimum level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.2.33 State**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
value: bool = driver.applications.k60Transient.calculate.deltaMarker.state.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTAmarker turns on delta marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>[:STATe]
driver.applications.k60Transient.calculate.deltaMarker.state.set(state = False,
↪window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTAmarker turns on delta marker 1.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.1.5.1.2.34 Trace

SCPI Commands

```
CALCulate<Window>:DELTAmarker<DeltaMarker>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTAmarker<m>:TRACe
value: float = driver.applications.k60Transient.calculate.deltaMarker.trace.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than ‘Blank’. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

trace: Trace number the marker is assigned to.

set(trace: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:TRACe
driver.applications.k60Transient.calculate.deltaMarker.trace.set(trace = 1.0,
↪window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

Trace number the marker is assigned to.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.5.1.2.35 X**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:X
```

class XCls

X commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:X
value: float = driver.applications.k60Transient.calculate.deltaMarker.x.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

position: Numeric value that defines the marker position on the x-axis. Range: The value range and unit depend on the measurement and scale of the x-axis.

set(position: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
driver.applications.k60Transient.calculate.deltaMarker.x.set(position = 1.0,
↪window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param position

Numeric value that defines the marker position on the x-axis. Range: The value range and unit depend on the measurement and scale of the x-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.deltaMarker.x.clone()
```

Subgroups

6.1.5.1.2.36 Relative

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:X:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X:RELative
value: float = driver.applications.k60Transient.calculate.deltaMarker.x.
↪relative.get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default)
```

This command queries the relative position of a delta marker on the x-axis. If necessary, the command activates the delta marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

position: Position of the delta marker in relation to the reference marker.

6.1.5.1.2.37 Y**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:Y
value: float = driver.applications.k60Transient.calculate.deltaMarker.y.
↳ get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

result: Result at the position of the delta marker. The unit is variable and depends on the one you have currently set. Unit: DBM

6.1.5.1.3 Distribution**class DistributionCls**

Distribution commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.distribution.clone()
```

Subgroups

6.1.5.1.3.1 Nbins

SCPI Commands

`CALCulate<Window>:DISTribution:NBINs`

class NbinsCls

Nbins commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → int

```
# SCPI: CALCulate<n>:DISTribution:NBINs
value: int = driver.applications.k60Transient.calculate.distribution.nbins.
↳get(window = repcap.Window.Default)
```

Defines the number of columns on the x-axis, i.e. the number of measurement value ranges for which the occurrences are determined.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

bins: Number of columns Range: 1 to 1000

set(*bins: int, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:DISTribution:NBINs
driver.applications.k60Transient.calculate.distribution.nbins.set(bins = 1,
↳window = repcap.Window.Default)
```

Defines the number of columns on the x-axis, i.e. the number of measurement value ranges for which the occurrences are determined.

param bins

Number of columns Range: 1 to 1000

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.4 HopDetection

class HopDetectionCls

HopDetection commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.hopDetection.clone()
```

Subgroups

6.1.5.1.4.1 Pnoise

class PnoiseCls

Pnoise commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.hopDetection.pnoise.clone()
```

Subgroups

6.1.5.1.4.2 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.hopDetection.pnoise.frequency.clone()
```

Subgroups

6.1.5.1.4.3 Start

SCPI Commands

```
CALCulate<Window>:HOPDetection:PNOise:FREQuency:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:FREQuency:STARt
value: float = driver.applications.k60Transient.calculate.hopDetection.pnoise.
    ↪ frequency.start.get(window = repcap.Window.Default)
```

Sets the start frequency for the phase noise hop measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

frequency: Unit: Hz

set(frequency: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:FREQuency:START
driver.applications.k60Transient.calculate.hopDetection.pnoise.frequency.start.
↪set(frequency = 1.0, window = repcap.Window.Default)
```

Sets the start frequency for the phase noise hop measurement.

param frequency

Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.4.4 Stop

SCPI Commands

```
CALCulate<Window>:HOPDetection:PNOise:FREQuency:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:FREQuency:STOP
value: float = driver.applications.k60Transient.calculate.hopDetection.pnoise.
↪frequency.stop.get(window = repcap.Window.Default)
```

Sets the stop frequency for the phase noise hop measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

frequency: Unit: Hz

set(frequency: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:FREQuency:STOP
driver.applications.k60Transient.calculate.hopDetection.pnoise.frequency.stop.
↪set(frequency = 1.0, window = repcap.Window.Default)
```

Sets the stop frequency for the phase noise hop measurement.

param frequency

Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.4.5 Length**SCPI Commands**

CALCulate<Window>:HOPDetection:PNOise:LENGth

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:LENGth
value: float = driver.applications.k60Transient.calculate.hopDetection.pnoise.
length.get(window = repcap.Window.Default)
```

Defines the length of the measurement range for power results in percent of the chirp length. This command is only available if the reference is CENT (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Reference. set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

percent: percent of the chirp length Range: 0 to 100

set(*percent: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:LENGth
driver.applications.k60Transient.calculate.hopDetection.pnoise.length.
set(percent = 1.0, window = repcap.Window.Default)
```

Defines the length of the measurement range for power results in percent of the chirp length. This command is only available if the reference is CENT (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Reference. set) .

param percent

percent of the chirp length Range: 0 to 100

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.4.6 Offset

class OffsetCls

Offset commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.hopDetection.pnoise.offset.clone()
```

Subgroups

6.1.5.1.4.7 Begin

SCPI Commands

```
CALCulate<Window>:HOPDetection:PNOise:OFFSet:BEgin
```

class BeginCls

Begin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:OFFSet:BEgin
value: float = driver.applications.k60Transient.calculate.hopDetection.pnoise.
↳offset.begin.get(window = repcap.Window.Default)
```

Defines the beginning of the measurement range for phase noise results as an offset in seconds from the hop start. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection. Pnoise.Reference.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

time: Unit: S

set(time: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:OFFSet:BEgin
driver.applications.k60Transient.calculate.hopDetection.pnoise.offset.begin.
↳set(time = 1.0, window = repcap.Window.Default)
```

Defines the beginning of the measurement range for phase noise results as an offset in seconds from the hop start. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection. Pnoise.Reference.set) .

param time

Unit: S

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.4.8 End

SCPI Commands

```
CALCulate<Window>:HOPDetection:PNOise:OFFSet:END
```

class EndCls

End commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:OFFSet:END
value: float = driver.applications.k60Transient.calculate.hopDetection.pnoise.
    offset.end.get(window = repcap.Window.Default)
```

Defines the end of the measurement range for phase noise results as an offset in seconds from the hop end. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise. Reference.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

time: Unit: S

set(*time: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:OFFSet:END
driver.applications.k60Transient.calculate.hopDetection.pnoise.offset.end.
    set(time = 1.0, window = repcap.Window.Default)
```

Defines the end of the measurement range for phase noise results as an offset in seconds from the hop end. This command is only available if the reference is EDGE (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise. Reference.set) .

param time

Unit: S

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.4.9 Reference

SCPI Commands

```
CALCulate<Window>:HOPDetection:PNOise:REference
```

class ReferenceCls

Reference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → ResultDevReference

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:REference
value: enums.ResultDevReference = driver.applications.k60Transient.calculate.
↳ hopDetection.pnoise.reference.get(window = repcap.Window.Default)
```

Defines the reference point for positioning the phase noise measurement range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

reference: CENTER | EDGE | FMSettling | PMSettling
EDGE The measurement range is defined in reference to the chirp’s rising or falling edge (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Offset.Begin.set and method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Offset.End.set) .
CENTER The measurement range is defined in reference to the center of the chirp. FMSettling The measurement range starts at the FM settling point (see [SENSe:]HOP:FMSettling:FMSPoint?) . PMSettling The measurement range starts at the PM settling point (see [SENSe:]HOP:PMSettling:PMSPoint?) .

set(*reference: ResultDevReference, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:HOPDetection:PNOise:REference
driver.applications.k60Transient.calculate.hopDetection.pnoise.reference.
↳ set(reference = enums.ResultDevReference.CENTER, window = repcap.Window.
↳ Default)
```

Defines the reference point for positioning the phase noise measurement range.

param reference

CENTER | EDGE | FMSettling | PMSettling
EDGE The measurement range is defined in reference to the chirp’s rising or falling edge (see method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Offset.Begin.set and method RsFswp.Applications.K60_Transient.Calculate.HopDetection.Pnoise.Offset.End.set) .
CENTER The measurement range is defined in reference to the center of the chirp. FMSettling The measurement range starts at the FM settling point (see [SENSe:]HOP:FMSettling:FMSPoint?) . PMSettling The measurement range starts at the PM settling point (see [SENSe:]HOP:PMSettling:PMSPoint?) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.5 Marker<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.calculate.marker.repcap_marker_get()
driver.applications.k60Transient.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 27 total commands, 10 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.clone()
```

Subgroups**6.1.5.1.5.1 Aoff****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:AOFF
driver.applications.k60Transient.calculate.marker.aoff.set(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command turns off all markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.2 LinkTo**class LinkToCls**

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.linkTo.clone()
```

Subgroups

6.1.5.1.5.3 Marker<MarkerDestination>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.calculate.marker.linkTo.marker.repcap_
↪markerDestination_get()
driver.applications.k60Transient.calculate.marker.linkTo.marker.repcap_markerDestination_
↪set(repcap.MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LINK:TO:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

get(window=Window.Default, marker=Marker.Default, markerDestination=MarkerDestination.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m1>:LINK:TO:MARKer<m2>
value: bool = driver.applications.k60Transient.calculate.marker.linkTo.marker.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default, ↪
↪markerDestination = repcap.MarkerDestination.Default)
```

This command links the normal source marker <ms> to any active destination marker <md> (normal or delta marker) . If you change the horizontal position of marker <md>, marker <ms> changes its horizontal position to the same value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m1>:LINK:TO:MARKer<m2>
driver.applications.k60Transient.calculate.marker.linkTo.marker.set(state =
↪False, window = repcap.Window.Default, marker = repcap.Marker.Default,
↪markerDestination = repcap.MarkerDestination.Default)
```

This command links the normal source marker <ms> to any active destination marker <md> (normal or delta marker). If you change the horizontal position of marker <md>, marker <ms> changes its horizontal position to the same value.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.linkTo.marker.clone()
```

6.1.5.1.5.4 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.maximum.clone()
```

Subgroups

6.1.5.1.5.5 Left

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:LEFT`

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:LEFT
driver.applications.k60Transient.calculate.marker.maximum.left.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.6 Next

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:NEXT
driver.applications.k60Transient.calculate.marker.maximum.next.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.7 Peak**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum[:PEAK]
driver.applications.k60Transient.calculate.marker.maximum.peak.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.8 Right**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MAXimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:RIGHT
driver.applications.k60Transient.calculate.marker.maximum.right.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.9 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.minimum.clone()
```

Subgroups

6.1.5.1.5.10 Left

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:LEFT
driver.applications.k60Transient.calculate.marker.minimum.left.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.11 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:NEXT
driver.applications.k60Transient.calculate.marker.minimum.next.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.12 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum[:PEAK]
driver.applications.k60Transient.calculate.marker.minimum.peak.set(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.13 Right

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:RIGHT
driver.applications.k60Transient.calculate.marker.minimum.right.set(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.1.5.14 Pexcursion

SCPI Commands

`CALCulate<Window>:MARKer:PEXCursion`

class PexcursionCls

Pexcursion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:MARKer:PEXCursion
value: float = driver.applications.k60Transient.calculate.marker.pexcursion.
↳get(window = repcap.Window.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

excursion: No help available

set(*excursion: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:MARKer:PEXCursion
driver.applications.k60Transient.calculate.marker.pexcursion.set(excursion = 1.
↪0, window = repcap.Window.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param excursion

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.5.15 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.clone()
```

Subgroups

6.1.5.1.5.16 Frame

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPEctrogram:FRAME
```

class FrameCls

Frame commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:FRAME
value: float = driver.applications.k60Transient.calculate.marker.spectrogram.
↪frame.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command positions a marker on a particular frame.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

frame_or_time: No help available

set(frame_or_time: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:FRAMe
driver.applications.k60Transient.calculate.marker.spectrogram.frame.set(frame_
↪or_time = 1.0, window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command positions a marker on a particular frame.

param frame_or_time

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.1.5.1.5.17 Sarea

SCPI Commands

```
CALCulate<Window>:MARKer:SPECTrogram:SARea
```

class SareaCls

Sarea commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → SearchArea

```
# SCPI: CALCulate<n>:MARKer:SPECTrogram:SARea
value: enums.SearchArea = driver.applications.k60Transient.calculate.marker.
↪spectrogram.sarea.get(window = repcap.Window.Default)
```

This command defines the marker search area for all spectrogram markers in the channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

search_area: VISible Performs a search within the visible frames. Note that the command does not work if the spectrogram is not visible for any reason (e.g. if the display update is off) . MEMory Performs a search within all frames in the memory.

set(search_area: SearchArea, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MARKer:SPECTrogram:SARea
driver.applications.k60Transient.calculate.marker.spectrogram.sarea.set(search_
↪area = enums.SearchArea.MEMory, window = repcap.Window.Default)
```

This command defines the marker search area for all spectrogram markers in the channel.

param search_area

VISible Performs a search within the visible frames. Note that the command does not work if the spectrogram is not visible for any reason (e.g. if the display update is off)
 . MEMory Performs a search within all frames in the memory.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.5.1.5.18 Xy**class XyCls**

Xy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.xy.clone()
```

Subgroups**6.1.5.1.5.19 Maximum****class MaximumCls**

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.xy.maximum.clone()
```

Subgroups**6.1.5.1.5.20 Peak****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:SPEctrogram:XY:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:XY:MAXimum[:PEAK]
driver.applications.k60Transient.calculate.marker.spectrogram.xy.maximum.peak.
↳ set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the highest level of the spectrogram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.21 Minimum

class MinimumCls

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.xy.minimum.clone()
```

Subgroups

6.1.5.1.5.22 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:XY:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:XY:MINimum[:PEAK]
driver.applications.k60Transient.calculate.marker.spectrogram.xy.minimum.peak.
    ↪ set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level of the spectrogram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.23 Y

class YCls

Y commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.y.clone()
```

Subgroups

6.1.5.1.5.24 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.y.maximum.clone()
```

Subgroups

6.1.5.1.5.25 Above

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:ABOVE
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MAXimum:ABOVE
driver.applications.k60Transient.calculate.marker.spectrogram.y.maximum.above.
    ↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next lower peak level for the current frequency. The search includes only frames above the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.26 Below

SCPI Commands

CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:BELOW

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MAXimum:BELOW
driver.applications.k60Transient.calculate.marker.spectrogram.y.maximum.below.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next lower peak level for the current frequency. The search includes only frames below the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.27 Next

SCPI Commands

CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:NEXT

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MAXimum:NEXT
driver.applications.k60Transient.calculate.marker.spectrogram.y.maximum.next.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next lower peak level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(*window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.5.28 Peak**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:SPEctrogram:Y:MAXimum:PEAK

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, marker=Marker.Default*) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:Y:MAXimum[:PEAK] driver.applications.k60Transient.calculate.marker.spectrogram.y.maximum.peak. ↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)</pre>

This command moves a marker vertically to the highest level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker. If the marker hasn't been active yet, the command looks for the peak level in the whole spectrogram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(*window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1*) → None

6.1.5.1.5.29 Minimum**class MinimumCls**

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.marker.spectrogram.y.minimum.clone()
```

Subgroups

6.1.5.1.5.30 Above

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:ABOVE
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:ABOVE
driver.applications.k60Transient.calculate.marker.spectrogram.y.minimum.above.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next higher minimum level for the current frequency. The search includes only frames above the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.31 Below

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:BELOW
driver.applications.k60Transient.calculate.marker.spectrogram.y.minimum.below.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next higher minimum level for the current frequency. The search includes only frames below the current marker position. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.32 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:NEXT
driver.applications.k60Transient.calculate.marker.spectrogram.y.minimum.next.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the next higher minimum level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.33 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum[:PEAK]
driver.applications.k60Transient.calculate.marker.spectrogram.y.minimum.peak.
↪set(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker vertically to the minimum level for the current frequency. The search includes all frames. It does not change the horizontal position of the marker. If the marker hasn't been active yet, the command first looks for the peak level for all frequencies and moves the marker vertically to the minimum level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.5.1.5.34 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
value: bool = driver.applications.k60Transient.calculate.marker.state.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
driver.applications.k60Transient.calculate.marker.state.set(state = False,
↪window = repcap.Window.Default, marker = repcap.Marker.Default)
```


This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.5.1.5.35 Trace

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
value: float = driver.applications.k60Transient.calculate.marker.trace.
↪ get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

trace: No help available

set(trace: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
driver.applications.k60Transient.calculate.marker.trace.set(trace = 1.0, window_
↪ = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.5.1.5.36 X**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:X

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X
value: float = driver.applications.k60Transient.calculate.marker.x.get(window = ↵
↵= repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

position: Numeric value that defines the marker position on the x-axis. The unit depends on the result display. Range: The range depends on the current x-axis range. , Unit: Hz

set(position: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X
driver.applications.k60Transient.calculate.marker.x.set(position = 1.0, window ↵
↵= repcap.Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param position

Numeric value that defines the marker position on the x-axis. The unit depends on the result display. Range: The range depends on the current x-axis range. , Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.5.1.5.37 Y

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.applications.k60Transient.calculate.marker.y.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: Unit: DBM

6.1.5.1.6 Spectrogram

SCPI Commands

```
CALCulate<Window>:SPECtrogram:CLEar
CALCulate<Window>:SPECtrogram:CLEar:ALL
```

class SpectrogramCls

Spectrogram commands group definition. 7 total commands, 3 Subgroups, 2 group commands

clear(window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram:CLEar
driver.applications.k60Transient.calculate.spectrogram.clear(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

clear_all(window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram:CLEar:ALL
driver.applications.k60Transient.calculate.spectrogram.clear_all(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

clear_all_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

clear_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.spectrogram.clone()
```

Subgroups

6.1.5.1.6.1 Frame

class FrameCls

Frame commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.spectrogram.frame.clone()
```

Subgroups

6.1.5.1.6.2 Count

SCPI Commands

```
CALCulate<Window>:SPECTrogram:FRAME:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → int

```
# SCPI: CALCulate<n>:SPECTrogram:FRAME:COUNT
value: int = driver.applications.k60Transient.calculate.spectrogram.frame.count.
↪ get(window = repcap.Window.Default)
```

This command queries the number of frames that are contained in the selected result display (depends on the evaluation basis) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

frames: The maximum number of frames depends on the history depth. Range: 1 to history depth

6.1.5.1.6.3 Select**SCPI Commands**

CALCulate<Window>:SPECtrogram:FRAME:SElect
--

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

<pre># SCPI: CALCulate<n>:SPECtrogram:FRAME:SElect value: float = driver.applications.k60Transient.calculate.spectrogram.frame. select.get(window = repcap.Window.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

frame_or_time: No help available

set(frame_or_time: float, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:SPECtrogram:FRAME:SElect driver.applications.k60Transient.calculate.spectrogram.frame.select.set(frame_ or_time = 1.0, window = repcap.Window.Default)</pre>
--

No command help available

param frame_or_time

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.6.4 Hdepth**SCPI Commands**

CALCulate<Window>:SPECtrogram:HDEPTH

class HdepthCls

Hdepth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:SPECtrogram:HDEPth
value: float = driver.applications.k60Transient.calculate.spectrogram.hdepth.
↪get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

depth: No help available

set(*depth: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECtrogram:HDEPth
driver.applications.k60Transient.calculate.spectrogram.hdepth.set(depth = 1.0, ↪
↪window = repcap.Window.Default)
```

No command help available

param depth

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.1.6.5 Tstamp

class TstampCls

Tstamp commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.calculate.spectrogram.tstamp.clone()
```

Subgroups

6.1.5.1.6.6 Data

SCPI Commands

```
CALCulate<Window>:SPECtrogram:TSTamp:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Seconds: float: No parameter help available
- Nanoseconds: float: No parameter help available
- Reserved: float: No parameter help available

get(frames: SelectionRangeB, window=Window.Default) → GetStruct

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp:DATA
value: GetStruct = driver.applications.k60Transient.calculate.spectrogram.
↳tstamp.data.get(frames = enums.SelectionRangeB.ALL, window = repcap.Window.
↳Default)
```

No command help available

param frames

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

structure: for return value, see the help for GetStruct structure arguments.

6.1.5.1.6.7 State**SCPI Commands**

CALCulate<Window>:SPECtrogram:TSTamp:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp[:STATe]
value: bool = driver.applications.k60Transient.calculate.spectrogram.tstamp.
↳state.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp[:STATe]
driver.applications.k60Transient.calculate.spectrogram.tstamp.state.set(state =
↳False, window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.5.2 Display

class DisplayCls

Display commands group definition. 13 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.clone()
```

Subgroups

6.1.5.2.1 Mtable

SCPI Commands

```
DISPlay:MTABle
```

class MtableCls

Mtable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoMode

```
# SCPI: DISPlay:MTABle
value: enums.AutoMode = driver.applications.k60Transient.display.mtable.get()
```

No command help available

return

display_mode: No help available

set(display_mode: AutoMode) → None

```
# SCPI: DISPlay:MTABle
driver.applications.k60Transient.display.mtable.set(display_mode = enums.
↳ AutoMode.AUTO)
```

No command help available

param display_mode

No help available

6.1.5.2.2 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.display.window.repcap_window_get()
driver.applications.k60Transient.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 11 total commands, 3 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.clone()
```

Subgroups

6.1.5.2.2.1 Size

SCPI Commands

```
DISPlay:WINDow<Window>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → Size

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
value: enums.Size = driver.applications.k60Transient.display.window.size.
↳ get(window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

size: LARGE Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

set(size: Size, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
driver.applications.k60Transient.display.window.size.set(size = enums.Size.
↳LARGE, window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` command (see method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set`).

param size

LARGE Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.5.2.2.2 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.spectrogram.clone()
```

Subgroups

6.1.5.2.2.3 Color

class ColorCls

Color commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.spectrogram.color.clone()
```

Subgroups

6.1.5.2.2.4 Style

SCPI Commands

```
DISPlay:WINDow<Window>:SPECTrogram:COLor:STYLE
```

class StyleCls

Style commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → ColorSchemeA

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor[:STYLE]
value: enums.ColorSchemeA = driver.applications.k60Transient.display.window.
    ↳ spectrogram.color.style.get(window = reprcap.Window.Default)
```

This command selects the color scheme. For details see ‘Color maps’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

return

color_scheme: HOT Uses a color range from blue to red. Blue colors indicate low levels, red colors indicate high ones. COLD Uses a color range from red to blue. Red colors indicate low levels, blue colors indicate high ones. RADar Uses a color range from black over green to light turquoise with shades of green in between. GRAYscale Shows the results in shades of gray.

set(color_scheme: ColorSchemeA, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor[:STYLE]
driver.applications.k60Transient.display.window.spectrogram.color.style.
    ↳ set(color_scheme = enums.ColorSchemeA.COLD, window = reprcap.Window.Default)
```

This command selects the color scheme. For details see ‘Color maps’.

param color_scheme

HOT Uses a color range from blue to red. Blue colors indicate low levels, red colors indicate high ones. COLD Uses a color range from red to blue. Red colors indicate low levels, blue colors indicate high ones. RADar Uses a color range from black over green to light turquoise with shades of green in between. GRAYscale Shows the results in shades of gray.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

6.1.5.2.2.5 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.applications.k60Transient.display.window.trace.repcap_trace_get()
driver.applications.k60Transient.display.window.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 9 total commands, 4 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.trace.clone()
```

Subgroups

6.1.5.2.2.6 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:MODE
```

class ModeCls

Mode commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TraceModeC

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
value: enums.TraceModeC = driver.applications.k60Transient.display.window.trace.
↳ mode.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: No help available

set(mode: TraceModeC, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
driver.applications.k60Transient.display.window.trace.mode.set(mode = enums.
↳TraceModeC.AVERage, window = repcap.Window.Default, trace = repcap.Trace.
↳Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.trace.mode.clone()
```

Subgroups

6.1.5.2.2.7 Hcontinuous

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:MODE:HCONtinuous
```

class HcontinuousCls

Hcontinuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE:HCONtinuous
value: bool = driver.applications.k60Transient.display.window.trace.mode.
↳hcontinuous.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE:HCONtinuous
driver.applications.k60Transient.display.window.trace.mode.hcontinuous.
↪set(state = False, window = repcap.Window.Default, trace = repcap.Trace.
↪Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.5.2.2.8 Preset

SCPI Commands

DISPlay:WINDow<Window>:TRACe<Trace>:PRESet

class PresetCls

Preset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TracePresetType

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
value: enums.TracePresetType = driver.applications.k60Transient.display.window.
↪trace.preset.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

result_type: No help available

set(result_type: TracePresetType, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
driver.applications.k60Transient.display.window.trace.preset.set(result_type =
↪enums.TracePresetType.ALL, window = repcap.Window.Default, trace = repcap.
↪Trace.Default)
```

No command help available

param result_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.5.2.2.9 State**SCPI Commands**

DISPlay:WINDow<Window>:TRACe<Trace>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

<pre># SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe] value: bool = driver.applications.k60Transient.display.window.trace.state. ↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(state: bool, window=Window.Default, trace=Trace.Default) → None

<pre># SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe] driver.applications.k60Transient.display.window.trace.state.set(state = False, ↳window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.5.2.2.10 Y

class YCls

Y commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.trace.y.clone()
```

Subgroups

6.1.5.2.2.11 Scale

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe
```

class ScaleCls

Scale commands group definition. 5 total commands, 3 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]
value: float = driver.applications.k60Transient.display.window.trace.y.scale.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

range_py: No help available

set(range_py: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]
driver.applications.k60Transient.display.window.trace.y.scale.set(range_py = 1.
↳0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param range_py

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.trace.y.scale.clone()
```

Subgroups**6.1.5.2.2.12 RefLevel****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe:Y:SCALe:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RLEVel
value: float = driver.applications.k60Transient.display.window.trace.y.scale.
↳ refLevel.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

reference_level: No help available

set(reference_level: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RLEVel
driver.applications.k60Transient.display.window.trace.y.scale.refLevel.
↳ set(reference_level = 1.0, window = repcap.Window.Default)
```

No command help available

param reference_level

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.display.window.trace.y.scale.refLevel.clone()
```

Subgroups

6.1.5.2.2.13 Offset

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe:Y:SCALE:RLEVel:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALE]:RLEVel:OFFSet
value: float = driver.applications.k60Transient.display.window.trace.y.scale.
↳refLevel.offset.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

offset: No help available

set(offset: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALE]:RLEVel:OFFSet
driver.applications.k60Transient.display.window.trace.y.scale.refLevel.offset.
↳set(offset = 1.0, window = repcap.Window.Default)
```

No command help available

param offset

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.5.2.2.14 RefPosition

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe:Y:SCALe:RPOSition
```

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RPOSition
value: float = driver.applications.k60Transient.display.window.trace.y.scale.
↪ refPosition.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

position: No help available

set(position: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RPOSition
driver.applications.k60Transient.display.window.trace.y.scale.refPosition.
↪ set(position = 1.0, window = repcap.Window.Default)
```

No command help available

param position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.5.2.2.15 Rvalue

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe:Y:SCALe:RVALue
```

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RVALue
value: float = driver.applications.k60Transient.display.window.trace.y.scale.
↪ rvalue.get(window = repcap.Window.Default)
```

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

value: numeric value WITHOUT UNIT Unit: dBm

set(value: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe:Y[:SCALe]:RVALue
driver.applications.k60Transient.display.window.trace.y.scale.rvalue.set(value_
↪= 1.0, window = repcap.Window.Default)
```

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param value

numeric value WITHOUT UNIT Unit: dBm

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.5.2.3 Wselect

SCPI Commands

DISPlay:WSElect

class WselectCls

Wselect commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: DISPlay:WSElect
value: int = driver.applications.k60Transient.display.wselect.get()
```

No command help available

return

active_window: No help available

6.1.5.3 FormatPy

class FormatPyCls

FormatPy commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.formatPy.clone()
```

Subgroups

6.1.5.3.1 Dexport

class DexportCls

Dexport commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.formatPy.dexport.clone()
```

Subgroups

6.1.5.3.1.1 Dseparator

SCPI Commands

```
FORMat:DEXPort:DSEPARATOR
```

class DseparatorCls

Dseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Separator

```
# SCPI: FORMat:DEXPort:DSEPARATOR
value: enums.Separator = driver.applications.k60Transient.formatPy.dexport.
↳ dseparator.get()
```

This command selects the decimal separator for data exported in ASCII format.

return

separator: POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05.
POINT Uses a point as decimal separator, e.g. 4.05.

set(separator: Separator) → None

```
# SCPI: FORMat:DEXPort:DSEPARATOR
driver.applications.k60Transient.formatPy.dexport.dseparator.set(separator =
↳ enums.Separator.COMMa)
```

This command selects the decimal separator for data exported in ASCII format.

param separator

POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05. POINT Uses
a point as decimal separator, e.g. 4.05.

6.1.5.3.1.2 Header

SCPI Commands

FORMat:DEXPort:HEADer

class HeaderCls

Header commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: FORMat:DEXPort:HEADer
value: bool = driver.applications.k60Transient.formatPy.dexport.header.get()
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

return
state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: FORMat:DEXPort:HEADer
driver.applications.k60Transient.formatPy.dexport.header.set(state = False)
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

param state
ON | OFF | 0 | 1

6.1.5.3.1.3 Traces

SCPI Commands

FORMat:DEXPort:TRACes

class TracesCls

Traces commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SelectionScope

```
# SCPI: FORMat:DEXPort:TRACes
value: enums.SelectionScope = driver.applications.k60Transient.formatPy.dexport.
↪traces.get()
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set) .

return
selection: SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

set(*selection: SelectionScope*) → None

```
# SCPI: FORMat:DEXPort:TRACes
driver.applications.k60Transient.formatPy.dexport.traces.set(selection = enums.
↳ SelectionScope.ALL)
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set).

param selection

SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

6.1.5.4 Initiate

class InitiateCls

Initiate commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.initiate.clone()
```

Subgroups

6.1.5.4.1 ConMeas

SCPI Commands

```
INITiate:CONMeas
```

class ConMeasCls

ConMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:CONMeas
driver.applications.k60Transient.initiate.conMeas.set()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:CONMeas
driver.applications.k60Transient.initiate.conMeas.set_with_opc()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.4.2 Continuous

SCPI Commands

INITiate:CONTinuous

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INITiate:CONTinuous
value: bool = driver.applications.k60Transient.initiate.continuous.get()
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

return

state: ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

set(state: bool) → None

```
# SCPI: INITiate:CONTinuous
driver.applications.k60Transient.initiate.continuous.set(state = False)
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

param state

ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

6.1.5.4.3 Immediate**SCPI Commands**

INITiate:IMMediate

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k60Transient.initiate.immediate.set()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. For details on synchronization see Remote control via SCPI.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k60Transient.initiate.immediate.set_with_opc()
```

This command starts a (single) new measurement. You can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. For details on synchronization see Remote control via SCPI.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.5 InputPy<InputIx>**RepCap Settings**

```
# Range: Nr1 .. Nr32
rc = driver.applications.k60Transient.inputPy.repcap_inputIx_get()
driver.applications.k60Transient.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 13 total commands, 9 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.clone()
```

Subgroups

6.1.5.5.1 Attenuation

SCPI Commands

```
INPut:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:ATTenuation
value: float = driver.applications.k60Transient.inputPy.attenuation.get()
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

return

attenuation: Range: see data sheet , Unit: DB

set(attenuation: float) → None

```
# SCPI: INPut:ATTenuation
driver.applications.k60Transient.inputPy.attenuation.set(attenuation = 1.0)
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

param attenuation

Range: see data sheet , Unit: DB

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.attenuation.clone()
```

Subgroups

6.1.5.5.1.1 Auto

SCPI Commands

```
INPut:ATTenuation:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:ATTenuation:AUTO
value: bool = driver.applications.k60Transient.inputPy.attenuation.auto.get()
```

This command couples or decouples the attenuation to the reference level. Thus, when the reference level is changed, the R&S FSWP determines the signal level for optimal internal data processing and sets the required attenuation accordingly.

```
return
state: ON | OFF | 0 | 1
```

set(state: bool) → None

```
# SCPI: INPut:ATTenuation:AUTO
driver.applications.k60Transient.inputPy.attenuation.auto.set(state = False)
```

This command couples or decouples the attenuation to the reference level. Thus, when the reference level is changed, the R&S FSWP determines the signal level for optimal internal data processing and sets the required attenuation accordingly.

```
param state
ON | OFF | 0 | 1
```

6.1.5.5.2 Connector

SCPI Commands

```
INPut:CONNector
```

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → InputConnectorC

```
# SCPI: INPut:CONNector
value: enums.InputConnectorC = driver.applications.k60Transient.inputPy.
connector.get()
```

No command help available

```
return
input_connectors: No help available
```

set(*input_connectors: InputConnectorC*) → None

```
# SCPI: INPut:CONNector
driver.applications.k60Transient.inputPy.connector.set(input_connectors = enums.
↳InputConnectorC.RF)
```

No command help available

param input_connectors

No help available

6.1.5.5.3 Coupling

SCPI Commands

```
INPut:COUPling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CouplingTypeA

```
# SCPI: INPut:COUPling
value: enums.CouplingTypeA = driver.applications.k60Transient.inputPy.coupling.
↳get()
```

This command selects the coupling type of the RF input.

return

coupling_type: AC | DC AC AC coupling DC DC coupling

set(*coupling_type: CouplingTypeA*) → None

```
# SCPI: INPut:COUPling
driver.applications.k60Transient.inputPy.coupling.set(coupling_type = enums.
↳CouplingTypeA.AC)
```

This command selects the coupling type of the RF input.

param coupling_type

AC | DC AC AC coupling DC DC coupling

6.1.5.5.4 Dpath

SCPI Commands

```
INPut:DPATH
```

class DpathCls

Dpath commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoOrOff

```
# SCPI: INPut:DPATH
value: enums.AutoOrOff = driver.applications.k60Transient.inputPy.dpath.get()
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

return
state: No help available

set(state: AutoOrOff) → None

```
# SCPI: INPut:DPATH
driver.applications.k60Transient.inputPy.dpath.set(state = enums.AutoOrOff.AUTO)
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

param state
No help available

6.1.5.5.5 Egain

class EgainCls

Egain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.egain.clone()
```

Subgroups

6.1.5.5.5.1 Bypass

SCPI Commands

```
INPut<InputIx>:EGain:BYPass
```

class BypassCls

Bypass commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:EGain:BYPass
value: bool = driver.applications.k60Transient.inputPy.egain.bypass.get(inputIx,
↳ repcap.InputIx.Default)
```

No command help available

param inputIx
optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:EGain:BYPass
driver.applications.k60Transient.inputPy.egain.bypass.set(state = False, ↵
↵ inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.5.5.2 State

SCPI Commands

```
INPut<InputIx>:EGain:STaTe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:EGain[:STaTe]
value: bool = driver.applications.k60Transient.inputPy.egain.state.get(inputIx, ↵
↵ = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:EGain[:STaTe]
driver.applications.k60Transient.inputPy.egain.state.set(state = False, inputIx, ↵
↵ = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.5.5.6 FilterPy

class FilterPyCls

FilterPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.filterPy.clone()
```

Subgroups

6.1.5.5.6.1 Hpass

class HpassCls

Hpass commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.filterPy.hpass.clone()
```

Subgroups

6.1.5.5.6.2 State

SCPI Commands

```
INPut:FILTer:HPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTer:HPASs[:STATe]
value: bool = driver.applications.k60Transient.inputPy.filterPy.hpass.state.
    ↪get()
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: INPut:FILTer:HPASs[:STATe]
driver.applications.k60Transient.inputPy.filterPy.hpass.state.set(state = False)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.5.5.6.3 Yig

class YigCls

Yig commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.filterPy.yig.clone()
```

Subgroups

6.1.5.5.6.4 State

SCPI Commands

```
INPut:FILTer:YIG:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTer:YIG[:STATe]
value: bool = driver.applications.k60Transient.inputPy.filterPy.yig.state.get()
```

Enables or disables the YIG filter.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: INPut:FILTer:YIG[:STATe]
driver.applications.k60Transient.inputPy.filterPy.yig.state.set(state = False)
```

Enables or disables the YIG filter.

param state
ON | OFF | 0 | 1

6.1.5.5.7 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.inputPy.gain.clone()
```

Subgroups

6.1.5.5.7.1 State

SCPI Commands

```
INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:GAIN:STATe
value: bool = driver.applications.k60Transient.inputPy.gain.state.get()
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: INPut:GAIN:STATe
driver.applications.k60Transient.inputPy.gain.state.set(state = False)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.5.5.7.2 Value

SCPI Commands

INPut:GAIN:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:GAIN[:VALue]
value: float = driver.applications.k60Transient.inputPy.gain.value.get()
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

return

gain: For R&S FSWP models 1322.8003K08, 1322.8003K09, 1322.8003K27 and 1322.8003K51, the following settings are available: 15 dB and 30 dB All other values are rounded to the nearest of these two. For R&S FSWP models 1322.8003K26 and 1322.8003K50: 30 dB Unit: DB

set(gain: float) → None

```
# SCPI: INPut:GAIN[:VALue]
driver.applications.k60Transient.inputPy.gain.value.set(gain = 1.0)
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param gain

For R&S FSWP models 1322.8003K08, 1322.8003K09, 1322.8003K27 and 1322.8003K51, the following settings are available: 15 dB and 30 dB All other values are rounded to the nearest of these two. For R&S FSWP models 1322.8003K26 and 1322.8003K50: 30 dB Unit: DB

6.1.5.5.8 Impedance

SCPI Commands

INPut:IMPedance

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: INPut:IMPedance
value: int = driver.applications.k60Transient.inputPy.impedance.get()
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

return
impedance: 50 | 75 Unit: OHM

set(impedance: int) → None

```
# SCPI: INPut:IMPedance
driver.applications.k60Transient.inputPy.impedance.set(impedance = 1)
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

param impedance
50 | 75 Unit: OHM

6.1.5.5.9 Select

SCPI Commands

INPut:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → InputSourceB

```
# SCPI: INPut:SElect
value: enums.InputSourceB = driver.applications.k60Transient.inputPy.select.
↳get()
```

This command selects the signal source for measurements, i.e. it defines which connector is used to input data to the R&S FSWP.

return
source: RF Radio Frequency ('RF INPUT' connector) FIQ I/Q data file (selected by method RsFswp.InputPy.File.Path.set) For details, see 'Basics on input from I/Q data files'.

set(source: InputSourceB) → None

```
# SCPI: INPut:SElect
driver.applications.k60Transient.inputPy.select.set(source = enums.InputSourceB.
↳FIQ)
```

This command selects the signal source for measurements, i.e. it defines which connector is used to input data to the R&S FSWP.

param source
RF Radio Frequency ('RF INPUT' connector) FIQ I/Q data file (selected by method RsFswp.InputPy.File.Path.set) For details, see 'Basics on input from I/Q data files'.

6.1.5.6 Layout

class LayoutCls

Layout commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.clone()
```

Subgroups

6.1.5.6.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.add.clone()
```

Subgroups

6.1.5.6.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK60) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k60Transient.layout.add.window.get(window_name_
↳ '1', direction = enums.WindowDirection.ABOVE, window_type = enums.
↳ WindowTypeK60.ChirpRateTimeDomain=CRTIME)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method RsFswp.Layout.Replace.Window.set command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_query**.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.5.6.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.catalog.clone()
```

Subgroups

6.1.5.6.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k60Transient.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return

result: No help available

6.1.5.6.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.identify.clone()
```

Subgroups

6.1.5.6.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k60Transient.layout.identify.window.get(window_
name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name

String containing the name of a window.

return

window_index: Index number of the window.

6.1.5.6.4 Remove

class RemoveCls

Remove commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.remove.clone()
```

Subgroups

6.1.5.6.4.1 Window

SCPI Commands

```
LAYout:REMove:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str) → None

```
# SCPI: LAYout:REMove[:WINDow]
driver.applications.k60Transient.layout.remove.window.set(window_name = '1')
```

This command removes a window from the display in the active channel.

param window_name

String containing the name of the window. In the default state, the name of the window is its index.

6.1.5.6.5 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.layout.replace.clone()
```

Subgroups

6.1.5.6.5.1 Window

SCPI Commands

```
LAYout:REPLace:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window_name*: str, *window_type*: WindowTypeK60) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k60Transient.layout.replace.window.set(window_name = '1',
↳window_type = enums.WindowTypeK60.ChirpRateTimeDomain=CRTime)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method RsFswp.Layout. **Add.Window.get_** command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method **RsFswp.Layout.Catalog.Window.get_** query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method **RsFswp.Layout.Add.Window.get_** for a list of available window types.

6.1.5.6.6 Splitter

SCPI Commands

LAYout:SPLitter

class SplitterCls

Splitter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*index_1*: int, *index_2*: int, *position*: int) → None

```
# SCPI: LAYout:SPLitter
driver.applications.k60Transient.layout.splitter.set(index_1 = 1, index_2 = 1,
↳position = 1)
```

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method RsFswp.Applications.K30_NoiseFigure.Display.Window.Size.set command, the method RsFswp. Applications.K30_NoiseFigure.Layout.Splitter.set changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

param index_1

The index of one window the splitter controls.

param index_2

The index of a window on the other side of the splitter.

param position

New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu) . The point of origin (x = 0, y = 0) is in the lower left corner of the screen. The end point (x = 100, y = 100) is in the upper right corner of the screen. (See Figure ‘SmartGrid coordinates for remote control of the splitters’.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

6.1.5.7 MassMemory

class MassMemoryCls

MassMemory commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.massMemory.clone()
```

Subgroups

6.1.5.7.1 Load

class LoadCls

Load commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.massMemory.load.clone()
```

Subgroups

6.1.5.7.1.1 Iq

class IqCls

Iq commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.massMemory.load.iq.clone()
```

Subgroups

6.1.5.7.1.2 Stream

SCPI Commands

```
MMEMemory:LOAD:IQ:STReam
```

class StreamCls

Stream commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:LOAD:IQ:STReam
value: str = driver.applications.k60Transient.massMemory.load.iq.stream.get()
```

No command help available

```
return
channel: No help available
```

set(channel: str) → None

```
# SCPI: MMEemory:LOAD:IQ:STReam
driver.applications.k60Transient.massMemory.load.iq.stream.set(channel = '1')
```

No command help available

```
param channel
No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.massMemory.load.iq.stream.clone()
```

Subgroups

6.1.5.7.1.3 Auto

SCPI Commands

```
MMEemory:LOAD:IQ:STReam:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEemory:LOAD:IQ:STReam:AUTO
value: bool = driver.applications.k60Transient.massMemory.load.iq.stream.auto.
↳get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: MMEemory:LOAD:IQ:STReam:AUTO
driver.applications.k60Transient.massMemory.load.iq.stream.auto.set(state =
↳False)
```

No command help available

param state
No help available

6.1.5.7.1.4 ListPy

SCPI Commands

```
MMEMory:LOAD:IQ:STReam:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: MMEMory:LOAD:IQ:STReam:LIST
value: List[str] = driver.applications.k60Transient.massMemory.load.iq.stream.
    ↪ listPy.get()
```

No command help available

return
result: No help available

6.1.5.7.2 Store<Store>

RepCap Settings

```
# Range: Pos1 .. Pos32
rc = driver.applications.k60Transient.massMemory.store.repcap_store_get()
driver.applications.k60Transient.massMemory.store.repcap_store_set(repcap.Store.Pos1)
```

class StoreCls

Store commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability:
Store, default value after init: Store.Pos1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.massMemory.store.clone()
```

Subgroups

6.1.5.7.2.1 Trace

SCPI Commands

```
MMEMory:STORe<Store>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(trace: int, filename: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:TRACe
driver.applications.k60Transient.massMemory.store.trace.set(trace = 1, filename_
↪ = '1', store = repcap.Store.Default)
```

This command exports trace data from the specified window to an ASCII file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a ‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device.

param trace

Number of the trace to be stored

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.1.5.8 Output<OutputConnector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.applications.k60Transient.output.repcap_outputConnector_get()
driver.applications.k60Transient.output.repcap_outputConnector_set(repcap.
↪ OutputConnector.Nr1)
```

class OutputCls

Output commands group definition. 7 total commands, 2 Subgroups, 0 group commands Repeated Capability: OutputConnector, default value after init: OutputConnector.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.output.clone()
```

Subgroups

6.1.5.8.1 Iqhs

class IqhsCls

Iqhs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.output.iqhs.clone()
```

Subgroups

6.1.5.8.1.1 Cdevice

SCPI Commands

```
OUTPut:IQHS:CDEvice
```

class CdeviceCls

Cdevice commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: OUTPut:IQHS:CDEvice
value: str = driver.applications.k60Transient.output.iqhs.cdevice.get()
```

No command help available

return
device: No help available

6.1.5.8.1.2 SymbolRate

SCPI Commands

```
OUTPut<OutputConnector>:IQHS:SRATe
```

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → float

```
# SCPI: OUTPut<up>:IQHS:SRATe
value: float = driver.applications.k60Transient.output.iqhs.symbolRate.
    get(outputConnector = repcap.OutputConnector.Default)
```

No command help available

param outputConnector
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return
sample_rate: No help available

6.1.5.8.2 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k60Transient.output.trigger.repcap_triggerPort_get()
driver.applications.k60Transient.output.trigger.repcap_triggerPort_set(repcap.
↳TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 4 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.output.trigger.clone()
```

Subgroups

6.1.5.8.2.1 Direction

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → InOutDirection

```
# SCPI: OUTPut:TRIGger<tp>:DIRection
value: enums.InOutDirection = driver.applications.k60Transient.output.trigger.
↳direction.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

direction: INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

set(direction: InOutDirection, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<tp>:DIRection
driver.applications.k60Transient.output.trigger.direction.set(direction = enums.
↳InOutDirection.INPut, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param direction

INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.5.8.2.2 Level**SCPI Commands**

```
OUTPut:TRIGger<TriggerPort>:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*triggerPort=TriggerPort.Default*) → LowHigh

```
# SCPI: OUTPut:TRIGger<tp>:LEVel
value: enums.LowHigh = driver.applications.k60Transient.output.trigger.level.
↳get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

level: HIGH 5 V LOW 0 V

set(*level: LowHigh, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut:TRIGger<tp>:LEVel
driver.applications.k60Transient.output.trigger.level.set(level = enums.LowHigh.
↳HIGH, triggerPort = repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param level

HIGH 5 V LOW 0 V

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.5.8.2.3 Otype

SCPI Commands

`OUTPut:TRIGger<TriggerPort>:OTYPE`

class OtypeCls

Otype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*triggerPort*=*TriggerPort.Default*) → *TriggerOutType*

```
# SCPI: OUTPut:TRIGger<tp>:OTYPE
value: enums.TriggerOutType = driver.applications.k60Transient.output.trigger.
↳otype.get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

output_type: DEvice Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEfined Sends a user-defined trigger signal. For more information, see method `Rsfswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set`.

set(*output_type*: *TriggerOutType*, *triggerPort*=*TriggerPort.Default*) → None

```
# SCPI: OUTPut:TRIGger<tp>:OTYPE
driver.applications.k60Transient.output.trigger.otype.set(output_type = enums.
↳TriggerOutType.DEvice, triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param output_type

DEvice Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEfined Sends a user-defined trigger signal. For more information, see method `Rsfswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set`.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.5.8.2.4 Pulse

class PulseCls

Pulse commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.output.trigger.pulse.clone()
```

Subgroups

6.1.5.8.2.5 Immediate

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:PULSe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:IMMediate
driver.applications.k60Transient.output.trigger.pulse.immediate.set(triggerPort,
↪ repcap.TriggerPort.Default)
```

This command generates a pulse at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

set_with_opc(triggerPort=TriggerPort.Default, opc_timeout_ms: int = -1) → None

6.1.5.8.2.6 Length

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:PULSe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:LENGth
value: float = driver.applications.k60Transient.output.trigger.pulse.length.
↪ get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

length: Pulse length in seconds. Unit: S

set(length: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<tp>:PULSe:LENGth
driver.applications.k60Transient.output.trigger.pulse.length.set(length = 1.0,
↪ triggerPort = repcap.TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param length

Pulse length in seconds. Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.5.9 Sense

class SenseCls

Sense commands group definition. 51 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.clone()
```

Subgroups

6.1.5.9.1 Adjust

class AdjustCls

Adjust commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.adjust.clone()
```

Subgroups

6.1.5.9.1.1 Level

SCPI Commands

```
SENSe:ADJust:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k60Transient.sense.adjust.level.set()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k60Transient.sense.adjust.level.set_with_opc()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.5.9.2 Correction

class CorrectionCls

Correction commands group definition. 11 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.correction.clone()
```

Subgroups

6.1.5.9.2.1 Cvl

SCPI Commands

```
SENSe:CORRection:CVL:CLEAr
```

class CvlCls

Cvl commands group definition. 11 total commands, 10 Subgroups, 1 group commands

clear() → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEAr
driver.applications.k60Transient.sense.correction.cvl.clear()
```

No command help available

clear_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEAr
driver.applications.k60Transient.sense.correction.cvl.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.correction.cvl.clone()
```

Subgroups

6.1.5.9.2.2 Band

SCPI Commands

```
SENSe:CORRection:CVL:BAND
```

class BandCls

Band commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → BandB

```
# SCPI: [SENSe]:CORRection:CVL:BAND
value: enums.BandB = driver.applications.k60Transient.sense.correction.cvl.band.
↳get()
```

No command help available

return

band: No help available

set(band: BandB) → None

```
# SCPI: [SENSe]:CORRection:CVL:BAND
driver.applications.k60Transient.sense.correction.cvl.band.set(band = enums.
↳BandB.D)
```

No command help available

param band

No help available

6.1.5.9.2.3 Bias

SCPI Commands

```
SENSe:CORRection:CVL:BIAS
```

class BiasCls

Bias commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
value: float = driver.applications.k60Transient.sense.correction.cvl.bias.get()
```

No command help available

return
bias_setting: No help available

set(bias_setting: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
driver.applications.k60Transient.sense.correction.cvl.bias.set(bias_setting = 1.
→0)
```

No command help available

param bias_setting
No help available

6.1.5.9.2.4 Catalog

SCPI Commands

```
SENSe:CORRection:CVL:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:CVL:CATalog
value: str = driver.applications.k60Transient.sense.correction.cvl.catalog.get()
```

No command help available

return
text: No help available

6.1.5.9.2.5 Comment

SCPI Commands

SENSe:CORRection:CVL:COMMeNt

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(text: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:COMMeNt
value: str = driver.applications.k60Transient.sense.correction.cv1.comment.
↳get(text = '1')
```

No command help available

param text

No help available

return

text: No help available

set(text: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:COMMeNt
driver.applications.k60Transient.sense.correction.cv1.comment.set(text = '1')
```

No command help available

param text

No help available

6.1.5.9.2.6 Data

SCPI Commands

SENSe:CORRection:CVL:DATA

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(freq: List[int], level: List[int]) → List[int]

```
# SCPI: [SENSe]:CORRection:CVL:DATA
value: List[int] = driver.applications.k60Transient.sense.correction.cv1.data.
↳get(freq = [1, 2, 3], level = [1, 2, 3])
```

No command help available

param freq

No help available

param level

No help available

return

freq: No help available

set(freq: List[int], level: List[int]) → None

```
# SCPI: [SENSe]:CORRection:CVL:DATA
driver.applications.k60Transient.sense.correction.cvl.data.set(freq = [1, 2, 3],
↪ level = [1, 2, 3])
```

No command help available

param freq

No help available

param level

No help available

6.1.5.9.2.7 Harmonic

SCPI Commands

SENSe:CORRection:CVL:HARMonic

class HarmonicCls

Harmonic commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(harm_order: float) → float

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
value: float = driver.applications.k60Transient.sense.correction.cvl.harmonic.
↪ get(harm_order = 1.0)
```

No command help available

param harm_order

No help available

return

harm_order: No help available

set(harm_order: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
driver.applications.k60Transient.sense.correction.cvl.harmonic.set(harm_order =
↪ 1.0)
```

No command help available

param harm_order

No help available

6.1.5.9.2.8 Mixer

SCPI Commands

SENSe:CORRection:CVL:MIXer

class MixerCls

Mixer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(type_py: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
value: str = driver.applications.k60Transient.sense.correction.cvl.mixer.
↳get(type_py = '1')
```

No command help available

param type_py
No help available

return
type_py: No help available

set(type_py: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
driver.applications.k60Transient.sense.correction.cvl.mixer.set(type_py = '1')
```

No command help available

param type_py
No help available

6.1.5.9.2.9 Ports

SCPI Commands

SENSe:CORRection:CVL:PORTs

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(port_type: int) → int

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
value: int = driver.applications.k60Transient.sense.correction.cvl.ports.
↳get(port_type = 1)
```

No command help available

param port_type
No help available

return
port_type: No help available

set(port_type: int) → None

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
driver.applications.k60Transient.sense.correction.cvl.ports.set(port_type = 1)
```

No command help available

param port_type
No help available

6.1.5.9.2.10 Select

SCPI Commands

```
SENSe:CORRection:CVL:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filename: str) → str

```
# SCPI: [SENSe]:CORRection:CVL:SElect
value: str = driver.applications.k60Transient.sense.correction.cvl.select.
    ↪get(filename = '1')
```

No command help available

param filename
No help available

return
filename: No help available

set(filename: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:SElect
driver.applications.k60Transient.sense.correction.cvl.select.set(filename = '1')
```

No command help available

param filename
No help available

6.1.5.9.2.11 Snumber

SCPI Commands

```
SENSe:CORRection:CVL:SNUMber
```

class SnumberCls

Snumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*serial_no: str*) → str

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
value: str = driver.applications.k60Transient.sense.correction.cvl.snumber.
↪ get(serial_no = '1')
```

No command help available

param serial_no

No help available

return

serial_no: No help available

set(*serial_no: str*) → None

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
driver.applications.k60Transient.sense.correction.cvl.snumber.set(serial_no = '1'
↪ )
```

No command help available

param serial_no

No help available

6.1.5.9.3 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.frequency.clone()
```

Subgroups

6.1.5.9.3.1 Center

SCPI Commands

```
SENSe:FREquency:CENTer
```

class CenterCls

Center commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREquency:CENTer
value: float = driver.applications.k60Transient.sense.frequency.center.get()
```

This command defines the center frequency.

return

frequency: The allowed range and fmax is specified in the data sheet. Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:CENTer
driver.applications.k60Transient.sense.frequency.center.set(frequency = 1.0)
```

This command defines the center frequency.

param frequency

The allowed range and fmax is specified in the data sheet. Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.frequency.center.clone()
```

Subgroups**6.1.5.9.3.2 Step****SCPI Commands**

```
SENSe:FREQuency:CENTer:STEP
```

class StepCls

Step commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP
value: float = driver.applications.k60Transient.sense.frequency.center.step.
    ↪get()
```

This command defines the center frequency step size.

return

stepsize: fmax is specified in the data sheet. Range: 1 to fMAX, Unit: Hz

set(stepsize: float) → None

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP
driver.applications.k60Transient.sense.frequency.center.step.set(stepsize = 1.0)
```

This command defines the center frequency step size.

param stepsize

fmax is specified in the data sheet. Range: 1 to fMAX, Unit: Hz

6.1.5.9.3.3 Offset

SCPI Commands

SENSe:FREQuency:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:OFFSet
value: float = driver.applications.k60Transient.sense.frequency.offset.get()
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type ‘frequency’ will be corrected for this shift numerically by the application. Note: In MSRA mode, the setting command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

return

offset: Range: -1 THz to 1 THz, Unit: HZ

set(offset: float) → None

```
# SCPI: [SENSe]:FREQuency:OFFSet
driver.applications.k60Transient.sense.frequency.offset.set(offset = 1.0)
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type ‘frequency’ will be corrected for this shift numerically by the application. Note: In MSRA mode, the setting command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

param offset

Range: -1 THz to 1 THz, Unit: HZ

6.1.5.9.4 Mixer

class MixerCls

Mixer commands group definition. 21 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.clone()
```

Subgroups

6.1.5.9.4.1 Bias

class BiasCls

Bias commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.bias.clone()
```

Subgroups

6.1.5.9.4.2 High

SCPI Commands

```
SENSe:MIxer:BIAS:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIxer:BIAS:HIGH
value: float = driver.applications.k60Transient.sense.mixer.bias.high.get()
```

No command help available

return

bias_setting: No help available

set(bias_setting: float) → None

```
# SCPI: [SENSe]:MIxer:BIAS:HIGH
driver.applications.k60Transient.sense.mixer.bias.high.set(bias_setting = 1.0)
```

No command help available

param bias_setting

No help available

6.1.5.9.4.3 Low

SCPI Commands

`SENSe:MIXer:BIAS:LOW`

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
value: float = driver.applications.k60Transient.sense.mixer.bias.low.get()
```

No command help available

return

 bias_setting: No help available

set(bias_setting: float) → None

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
driver.applications.k60Transient.sense.mixer.bias.low.set(bias_setting = 1.0)
```

No command help available

param bias_setting

 No help available

6.1.5.9.4.4 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.frequency.clone()
```

Subgroups

6.1.5.9.4.5 Handover

SCPI Commands

`SENSe:MIXer:FREQuency:HANdOver`

class HandoverCls

Handover commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:HANdOver
value: float = driver.applications.k60Transient.sense.mixer.frequency.handover.
↳get()
```

No command help available

```
return
    handover: No help available
```

set(handover: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:HANdOver
driver.applications.k60Transient.sense.mixer.frequency.handover.set(handover =
↳1.0)
```

No command help available

```
param handover
    No help available
```

6.1.5.9.4.6 Start

SCPI Commands

```
SENSe:MIXer:FREQuency:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:START
value: float = driver.applications.k60Transient.sense.mixer.frequency.start.
↳get()
```

No command help available

```
return
    frequency: No help available
```

6.1.5.9.4.7 Stop

SCPI Commands

```
SENSe:MIXer:FREQuency:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:STOP
value: float = driver.applications.k60Transient.sense.mixer.frequency.stop.get()
```

No command help available

```
return
frequency: No help available
```

6.1.5.9.4.8 Harmonic

class HarmonicCls

Harmonic commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.harmonic.clone()
```

Subgroups

6.1.5.9.4.9 Band

SCPI Commands

```
SENSe:MIXer:HARMonic:BAND
SENSe:MIXer:HARMonic:BAND:PRESet
```

class BandCls

Band commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get() → BandB

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
value: enums.BandB = driver.applications.k60Transient.sense.mixer.harmonic.band.
↳get()
```

No command help available

```
return
band: No help available
```

preset() → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.applications.k60Transient.sense.mixer.harmonic.band.preset()
```

No command help available

preset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.applications.k60Transient.sense.mixer.harmonic.band.preset_with_opc()
```

No command help available

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

set(band: BandB) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
driver.applications.k60Transient.sense.mixer.harmonic.band.set(band = enums.
↳ BandB.D)
```

No command help available

param band

No help available

6.1.5.9.4.10 High

class HighCls

High commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.harmonic.high.clone()
```

Subgroups

6.1.5.9.4.11 State

SCPI Commands

```
SENSe:MIXer:HARMonic:HIGH:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
value: bool = driver.applications.k60Transient.sense.mixer.harmonic.high.state.
↳ get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
driver.applications.k60Transient.sense.mixer.harmonic.high.state.set(state =
↪False)
```

No command help available

param state

No help available

6.1.5.9.4.12 Value

SCPI Commands

```
SENSe:MIXer:HARMonic:HIGH:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH[:VALue]
value: float = driver.applications.k60Transient.sense.mixer.harmonic.high.value.
↪get()
```

No command help available

return

harm_order: No help available

set(harm_order: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH[:VALue]
driver.applications.k60Transient.sense.mixer.harmonic.high.value.set(harm_order
↪= 1.0)
```

No command help available

param harm_order

No help available

6.1.5.9.4.13 Low

SCPI Commands

```
SENSe:MIXer:HARMonic:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
value: float = driver.applications.k60Transient.sense.mixer.harmonic.low.get()
```

No command help available

```
return
    harm_order: No help available
```

set(harm_order: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
driver.applications.k60Transient.sense.mixer.harmonic.low.set(harm_order = 1.0)
```

No command help available

```
param harm_order
    No help available
```

6.1.5.9.4.14 TypePy

SCPI Commands

```
SENSe:MIXer:HARMonic:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → OddEven

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
value: enums.OddEven = driver.applications.k60Transient.sense.mixer.harmonic.
↳ typePy.get()
```

No command help available

```
return
    odd_even: No help available
```

set(odd_even: OddEven) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
driver.applications.k60Transient.sense.mixer.harmonic.typePy.set(odd_even =
↳ enums.OddEven.EODD)
```

No command help available

```
param odd_even
    No help available
```

6.1.5.9.4.15 LoPower

SCPI Commands

SENSe:MIXer:LOPower

class LoPowerCls

LoPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOPower
value: float = driver.applications.k60Transient.sense.mixer.loPower.get()
```

No command help available

return

level: No help available

set(level: float) → None

```
# SCPI: [SENSe]:MIXer:LOPower
driver.applications.k60Transient.sense.mixer.loPower.set(level = 1.0)
```

No command help available

param level

No help available

6.1.5.9.4.16 Loss

class LossCls

Loss commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.loss.clone()
```

Subgroups

6.1.5.9.4.17 High

SCPI Commands

SENSe:MIXer:LOSS:HIGH

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
value: float = driver.applications.k60Transient.sense.mixer.loss.high.get()
```

No command help available

return
average: No help available

set(average: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
driver.applications.k60Transient.sense.mixer.loss.high.set(average = 1.0)
```

No command help available

param average
No help available

6.1.5.9.4.18 Low

SCPI Commands

```
SENSe:MIXer:LOSS:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
value: float = driver.applications.k60Transient.sense.mixer.loss.low.get()
```

No command help available

return
average: No help available

set(average: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
driver.applications.k60Transient.sense.mixer.loss.low.set(average = 1.0)
```

No command help available

param average
No help available

6.1.5.9.4.19 Table

class TableCls

Table commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.loss.table.clone()
```

Subgroups

6.1.5.9.4.20 High

SCPI Commands

```
SENSe:MiXer:LOSS:TABLE:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filename: str) → str

```
# SCPI: [SENSe]:MiXer:LOSS:TABLE:HIGH
value: str = driver.applications.k60Transient.sense.mixer.loss.table.high.
↳get(filename = '1')
```

No command help available

param filename

No help available

return

filename: No help available

set(filename: str) → None

```
# SCPI: [SENSe]:MiXer:LOSS:TABLE:HIGH
driver.applications.k60Transient.sense.mixer.loss.table.high.set(filename = '1')
```

No command help available

param filename

No help available

6.1.5.9.4.21 Low

SCPI Commands

```
SENSe:MIXer:LOSS:TABLE:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filename: str) → str

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
value: str = driver.applications.k60Transient.sense.mixer.loss.table.low.
    get(filename = '1')
```

No command help available

param filename

No help available

return

filename: No help available

set(filename: str) → None

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
driver.applications.k60Transient.sense.mixer.loss.table.low.set(filename = '1')
```

No command help available

param filename

No help available

6.1.5.9.4.22 Ports

SCPI Commands

```
SENSe:MIXer:PORTs
```

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:MIXer:PORTs
value: int = driver.applications.k60Transient.sense.mixer.ports.get()
```

No command help available

return

port_type: No help available

set(port_type: int) → None

```
# SCPI: [SENSe]:MIXer:PORTs
driver.applications.k60Transient.sense.mixer.ports.set(port_type = 1)
```

No command help available

param port_type
No help available

6.1.5.9.4.23 RfOvrrange

class RfOvrrangeCls

RfOvrrange commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.mixer.rfOvrrange.clone()
```

Subgroups

6.1.5.9.4.24 State

SCPI Commands

```
SENSe:MIXer:RFOVrrange:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:RFOVrrange[:STATE]
value: bool = driver.applications.k60Transient.sense.mixer.rfOvrrange.state.
↳get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:MIXer:RFOVrrange[:STATE]
driver.applications.k60Transient.sense.mixer.rfOvrrange.state.set(state =
↳False)
```

No command help available

param state
No help available

6.1.5.9.4.25 Signal

SCPI Commands

```
SENSe:MIXer:SIGNa1
```

class SignalCls

Signal commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → State

```
# SCPI: [SENSe]:MIXer:SIGNa1
value: enums.State = driver.applications.k60Transient.sense.mixer.signal.get()
```

No command help available

```
return
    state: No help available
```

set(state: State) → None

```
# SCPI: [SENSe]:MIXer:SIGNa1
driver.applications.k60Transient.sense.mixer.signal.set(state = enums.State.ALL)
```

No command help available

```
param state
    No help available
```

6.1.5.9.4.26 State

SCPI Commands

```
SENSe:MIXer:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer[:STATe]
value: bool = driver.applications.k60Transient.sense.mixer.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: [SENSe]:MIXer[:STATe]
driver.applications.k60Transient.sense.mixer.state.set(state = False)
```

No command help available

param state
No help available

6.1.5.9.4.27 Threshold

SCPI Commands

SENSe:MIXer:THReshold

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:THReshold
value: float = driver.applications.k60Transient.sense.mixer.threshold.get()
```

No command help available

return
value: No help available

set(value: float) → None

```
# SCPI: [SENSe]:MIXer:THReshold
driver.applications.k60Transient.sense.mixer.threshold.set(value = 1.0)
```

No command help available

param value
No help available

6.1.5.9.5 Msra

class MsraCls

Msra commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.msra.clone()
```

Subgroups

6.1.5.9.5.1 Capture

class CaptureCls

Capture commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.msra.capture.clone()
```

Subgroups

6.1.5.9.5.2 Offset

SCPI Commands

```
SENSe:MSRA:CAPTure:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MSRA:CAPTure:OFFSet
value: float = driver.applications.k60Transient.sense.msra.capture.offset.get()
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

return

offset: This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

set(offset: float) → None

```
# SCPI: [SENSe]:MSRA:CAPTure:OFFSet
driver.applications.k60Transient.sense.msra.capture.offset.set(offset = 1.0)
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

param offset

This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

6.1.5.9.6 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k60Transient.sense.probe.repcap_probe_get()
driver.applications.k60Transient.sense.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 12 total commands, 2 Subgroups, 0 group commands Repeated Capability:
Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.probe.clone()
```

Subgroups

6.1.5.9.6.1 Id

class IdCls

Id commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.probe.id.clone()
```

Subgroups

6.1.5.9.6.2 PartNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:PARTnumber
```

class PartNumberCls

PartNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:ID:PARTnumber
value: float = driver.applications.k60Transient.sense.probe.id.partNumber.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

part_number: No help available

6.1.5.9.6.3 SrNumber**SCPI Commands**

```
SENSe:PROBe<Probe>:ID:SRNumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → str

```
# SCPI: [SENSe]:PROBe<pb>:ID:SRNumber
value: str = driver.applications.k60Transient.sense.probe.id.srNumber.get(probe_
↳ = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

serial_no: No help available

6.1.5.9.6.4 Setup**class SetupCls**

Setup commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.probe.setup.clone()
```

Subgroups**6.1.5.9.6.5 AttRatio****SCPI Commands**

```
SENSe:PROBe<Probe>:SETup:ATTRatio
```

class AttRatioCls

AttRatio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
value: float = driver.applications.k60Transient.sense.probe.setup.attRatio.
↪get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

attenuation_ratio: No help available

set(*attenuation_ratio: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
driver.applications.k60Transient.sense.probe.setup.attRatio.set(attenuation_
↪ratio = 1.0, probe = repcap.Probe.Default)
```

No command help available

param attenuation_ratio

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.6 CmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:CMOffset
```

class CmOffsetCls

CmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
value: float = driver.applications.k60Transient.sense.probe.setup.cmOffset.
↪get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

cm_offset: No help available

set(*cm_offset*: float, *probe*=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
driver.applications.k60Transient.sense.probe.setup.cmOffset.set(cm_offset = 1.0,
↪ probe = repcap.Probe.Default)
```

No command help available

param cm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.7 DmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:DMOffset
```

class DmOffsetCls

DmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe*=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOffset
value: float = driver.applications.k60Transient.sense.probe.setup.dmOffset.
↪ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

dm_offset: No help available

set(*dm_offset*: float, *probe*=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOffset
driver.applications.k60Transient.sense.probe.setup.dmOffset.set(dm_offset = 1.0,
↪ probe = repcap.Probe.Default)
```

No command help available

param dm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.8 Mode

SCPI Commands

SENSe:PROBe<Probe>:SETup:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeSetupMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
value: enums.ProbeSetupMode = driver.applications.k60Transient.sense.probe.
↳ setup.mode.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(*mode: ProbeSetupMode, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
driver.applications.k60Transient.sense.probe.setup.mode.set(mode = enums.
↳ ProbeSetupMode.NOAction, probe = repcap.Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.9 Name

SCPI Commands

SENSe:PROBe<Probe>:SETup:NAME

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → str

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NAME
value: str = driver.applications.k60Transient.sense.probe.setup.name.get(probe,
↳ repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

name: No help available

6.1.5.9.6.10 NmOffset**SCPI Commands**

SENSe:PROBe<Probe>:SETup:NMOffset

class NmOffsetCls

NmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

<pre># SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset value: float = driver.applications.k60Transient.sense.probe.setup.nmOffset. ↳ get(probe = repcap.Probe.Default)</pre>

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

nm_offset: No help available

set(*nm_offset: float, probe=Probe.Default*) → None

<pre># SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset driver.applications.k60Transient.sense.probe.setup.nmOffset.set(nm_offset = 1.0, ↳ probe = repcap.Probe.Default)</pre>
--

No command help available

param nm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.11 Pmode

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMODE
```

class PmodeCls

Pmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
value: enums.ProbeMode = driver.applications.k60Transient.sense.probe.setup.
↳ pmode.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(*mode: ProbeMode, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
driver.applications.k60Transient.sense.probe.setup.pmode.set(mode = enums.
↳ ProbeMode.CM, probe = repcap.Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.12 PmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMOffset
```

class PmOffsetCls

PmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
value: float = driver.applications.k60Transient.sense.probe.setup.pmOffset.
↳ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

pm_offset: No help available

set(pm_offset: float, probe=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
driver.applications.k60Transient.sense.probe.setup.pmOffset.set(pm_offset = 1.0,
↪ probe = repcap.Probe.Default)
```

No command help available

param pm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.1.5.9.6.13 State**SCPI Commands**

```
SENSe:PROBe<Probe>:SETup:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → Detect

```
# SCPI: [SENSe]:PROBe<pb>:SETup:STATe
value: enums.Detect = driver.applications.k60Transient.sense.probe.setup.state.
↪ get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

state: No help available

6.1.5.9.6.14 TypePy

SCPI Commands

`SENSe:PROBe<Probe>:SETup:TYPE`

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(probe=Probe.Default) → str`

```
# SCPI: [SENSe]:PROBe<pb>:SETup:TYPE
value: str = driver.applications.k60Transient.sense.probe.setup.typePy.
↳get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

type_py: No help available

6.1.5.9.7 Sweep

class SweepCls

Sweep commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.sweep.clone()
```

Subgroups

6.1.5.9.7.1 Count

SCPI Commands

`SENSe:SWEEp:COUNT`

class CountCls

Count commands group definition. 2 total commands, 1 Subgroups, 1 group commands

`get() → float`

```
# SCPI: [SENSe]:SWEEp:COUNT
value: float = driver.applications.k60Transient.sense.sweep.count.get()
```

This command defines the number of measurements that the application uses to average traces. In continuous measurement mode, the application calculates the moving average over the average count. In single measurement mode, the application stops the measurement and calculates the average after the average count has been reached.

return
sweep_count: No help available

set(sweep_count: float) → None

```
# SCPI: [SENSe]:SWEep:COUNT
driver.applications.k60Transient.sense.sweep.count.set(sweep_count = 1.0)
```

This command defines the number of measurements that the application uses to average traces. In continuous measurement mode, the application calculates the moving average over the average count. In single measurement mode, the application stops the measurement and calculates the average after the average count has been reached.

param sweep_count
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.sense.sweep.count.clone()
```

Subgroups

6.1.5.9.7.2 Current

SCPI Commands

```
SENSe:SWEep:COUNT:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:SWEep:COUNT:CURRent
value: int = driver.applications.k60Transient.sense.sweep.count.current.get()
```

This query returns the current number of started sweeps or measurements. This command is only available if a sweep count value is defined and the instrument is in single sweep mode.

return
value: No help available

6.1.5.10 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k60Transient.trace.repcap_window_get()
driver.applications.k60Transient.trace.repcap_window_set(repcap.Window.Nr1)
```

class TraceCls

Trace commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trace.clone()
```

Subgroups

6.1.5.10.1 Data

SCPI Commands

```
FORMAT REAL,32;TRACe<n>:DATA
```

class DataCls

Data commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(trace_type: TraceTypeK60, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.applications.k60Transient.trace.data.get(trace_type,
↪= enums.TraceTypeK60.SGRam, window = repcap.Window.Default)
```

No command help available

param trace_type
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return
trace_ydata: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trace.data.clone()
```

Subgroups

6.1.5.10.1.1 X

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_type: TraceTypeK60, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]:X
value: List[float] = driver.applications.k60Transient.trace.data.x.get(trace_
    ↪type = enums.TraceTypeK60.SGRam, window = repcap.Window.Default)
```

This remote control command returns the X values only for the trace in the selected result display. Depending on the type of result display and the scaling of the x-axis, this can be either the pulse number or a timestamp for each detected pulse in the capture buffer. This command is only available for graphical displays, except for the Magnitude Capture display.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_xdata: No help available

6.1.5.11 Trigger

class TriggerCls

Trigger commands group definition. 10 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.clone()
```

Subgroups

6.1.5.11.1 Sequence

class SequenceCls

Sequence commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.sequence.clone()
```

Subgroups

6.1.5.11.1.1 Dtime

SCPI Commands

```
TRIGger:SEquence:DTIME
```

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:DTIME
value: float = driver.applications.k60Transient.trigger.sequence.dtime.get()
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

return

dropout_time: Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

set(dropout_time: float) → None

```
# SCPI: TRIGger[:SEquence]:DTIME
driver.applications.k60Transient.trigger.sequence.dtime.set(dropout_time = 1.0)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param dropout_time

Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

6.1.5.11.1.2 Holdoff

class HoldoffCls

Holdoff commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.sequence.holdoff.clone()
```

Subgroups

6.1.5.11.1.3 Time

SCPI Commands

```
TRIGger:SEquence:HOLDoff:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
value: float = driver.applications.k60Transient.trigger.sequence.holdoff.time.
    ↪get()
```

Defines the time offset between the trigger event and the start of the measurement.

return
offset: Unit: S

set(offset: float) → None

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
driver.applications.k60Transient.trigger.sequence.holdoff.time.set(offset = 1.0)
```

Defines the time offset between the trigger event and the start of the measurement.

param offset
Unit: S

6.1.5.11.1.4 IfPower

class IfPowerCls

IfPower commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.sequence.ifPower.clone()
```

Subgroups

6.1.5.11.1.5 Holdoff

SCPI Commands

```
TRIGger:SEquence:IFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
value: float = driver.applications.k60Transient.trigger.sequence.ifPower.
↳holdoff.get()
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) .

return

period: Range: 0 s to 10 s, Unit: S

set(period: float) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
driver.applications.k60Transient.trigger.sequence.ifPower.holdoff.set(period =
↳1.0)
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) .

param period

Range: 0 s to 10 s, Unit: S

6.1.5.11.1.6 Hysteresis

SCPI Commands

```
TRIGger:SEquence:IFPower:HYSTeresis
```

class HysteresisCls

Hysteresis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
value: float = driver.applications.k60Transient.trigger.sequence.ifPower.
↳hysteresis.get()
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

return
hysteresis: Range: 3 dB to 50 dB, Unit: DB

set(hysteresis: float) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
driver.applications.k60Transient.trigger.sequence.ifPower.hysteresis.
↳set(hysteresis = 1.0)
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

param hysteresis
Range: 3 dB to 50 dB, Unit: DB

6.1.5.11.1.7 Level

class LevelCls

Level commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.sequence.level.clone()
```

Subgroups

6.1.5.11.1.8 External<ExternalPort>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.applications.k60Transient.trigger.sequence.level.external.repcap_
↳externalPort_get()
driver.applications.k60Transient.trigger.sequence.level.external.repcap_externalPort_
↳set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
TRIGger:SEquence:LEVel:EXternal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(externalPort=ExternalPort.Default) → float

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXternal<port>]
value: float = driver.applications.k60Transient.trigger.sequence.level.external.
↳get(externalPort = repcap.ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

trigger_level: Range: 0.5 V to 3.5 V, Unit: V

set(trigger_level: float, externalPort=ExternalPort.Default) → None

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXternal<port>]
driver.applications.k60Transient.trigger.sequence.level.external.set(trigger_
↳level = 1.0, externalPort = repcap.ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param trigger_level

Range: 0.5 V to 3.5 V, Unit: V

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k60Transient.trigger.sequence.level.external.clone()
```

6.1.5.11.1.9 IfPower

SCPI Commands

```
TRIGger:SEquence:LEVel:IFPower
```

class IfPowerCls

IfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
value: float = driver.applications.k60Transient.trigger.sequence.level.ifPower.
↪get()
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

return

trigger_level: For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

set(trigger_level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
driver.applications.k60Transient.trigger.sequence.level.ifPower.set(trigger_
↪level = 1.0)
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param trigger_level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.1.5.11.1.10 IqPower

SCPI Commands

TRIGger:SEquence:LEVel:IQPower

class IqPowerCls

IqPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
value: float = driver.applications.k60Transient.trigger.sequence.level.iqPower.
↪get()
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

return

trigger_level: Range: -130 dBm to 30 dBm, Unit: DBM

set(trigger_level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
driver.applications.k60Transient.trigger.sequence.level.iqPower.set(trigger_
↪level = 1.0)
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param trigger_level

Range: -130 dBm to 30 dBm, Unit: DBM

6.1.5.11.11 RfPower

SCPI Commands

TRIGger:SEquence:LEVel:RFPower

class RfPowerCls

RfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:RFPower
value: float = driver.applications.k60Transient.trigger.sequence.level.rfPower.
↳get()
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

return

trigger_level: For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

set(trigger_level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:RFPower
driver.applications.k60Transient.trigger.sequence.level.rfPower.set(trigger_
↳level = 1.0)
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

param trigger_level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.1.5.11.12 Slope

SCPI Commands

TRIGger:SEquence:SLOPe

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SlopeType

```
# SCPI: TRIGger[:SEquence]:SLOPe
value: enums.SlopeType = driver.applications.k60Transient.trigger.sequence.
↪slope.get()
```

This command selects the trigger slope.

return

type_py: POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

set(type_py: SlopeType) → None

```
# SCPI: TRIGger[:SEquence]:SLOPe
driver.applications.k60Transient.trigger.sequence.slope.set(type_py = enums.
↪SlopeType.NEGative)
```

This command selects the trigger slope.

param type_py

POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

6.1.5.11.1.13 Source

SCPI Commands

TRIGger:SEquence:SOURce

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerSourceK

```
# SCPI: TRIGger[:SEquence]:SOURce
value: enums.TriggerSourceK = driver.applications.k60Transient.trigger.sequence.
↪source.get()
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

return

source: IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’. IFPower Second intermediate frequency IQPower Magnitude of sampled I/Q data For applications that process I/Q data, such as the I/Q Analyzer or optional applications.

set(source: TriggerSourceK) → None

```
# SCPI: TRIGger[:SEquence]:SOURce
driver.applications.k60Transient.trigger.sequence.source.set(source = enums.
↳ TriggerSourceK.EXT2)
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

param source

IMMediate Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’. IFPower Second intermediate frequency IQPower Magnitude of sampled I/Q data For applications that process I/Q data, such as the I/Q Analyzer or optional applications.

6.1.5.12 TriggerInvoke

SCPI Commands

```
*TRG
```

class TriggerInvokeCls

TriggerInvoke commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: *TRG
driver.applications.k60Transient.triggerInvoke.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: *TRG
driver.applications.k60Transient.triggerInvoke.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.6 K7_AnalogDemod

class K7_AnalogDemodCls

K7_AnalogDemod commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k7_AnalogDemod.clone()
```

Subgroups

6.1.6.1 Layout

class LayoutCls

Layout commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k7AnalogDemod.layout.clone()
```

Subgroups

6.1.6.1.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k7AnalogDemod.layout.add.clone()
```

Subgroups

6.1.6.1.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK7) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k7AnalogDemod.layout.add.window.get(window_
↳ name = '1', direction = enums.WindowDirection.AB0Ve, window_type = enums.
↳ WindowTypeK7.AmSpectrum='XTIM:AM:RELative:AFSPpectrum')
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method `RsFswp.Layout.Replace.Window.set` command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **`RsFswp.Layout.Catalog.Window.get_query`**.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

`new_window_name`: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.6.1.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k7AnalogDemod.layout.catalog.clone()
```

Subgroups

6.1.6.1.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get()` → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k7AnalogDemod.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

```

return
    result: No help available

```

6.1.6.1.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```

# Create a clone of the original group, that exists independently
group2 = driver.applications.k7AnalogDemod.layout.identify.clone()

```

Subgroups

6.1.6.1.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get(window_name: str) → int
```

```

# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k7AnalogDemod.layout.identify.window.
    ↪get(window_name = '1')

```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name
String containing the name of a window.

return
window_index: Index number of the window.

6.1.6.1.4 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k7AnalogDemod.layout.replace.clone()
```

Subgroups

6.1.6.1.4.1 Window

SCPI Commands

```
LAYout:REPLace:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeK7) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k7AnalogDemod.layout.replace.window.set(window_name = '1',
↳window_type = enums.WindowTypeK7.AmSpectrum='XTIM:AM:RELative:AFSPpectrum')
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method RsFswp.Layout. **Add.Window.get_** command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method **RsFswp.Layout.Catalog.Window.get_** query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method **RsFswp.Layout.Add.Window.get_** for a list of available window types.

6.1.7 K70_Vsa

class K70_VsaCls

K70_Vsa commands group definition. 466 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70_Vsa.clone()
```

Subgroups

6.1.7.1 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.calculate.repcap_window_get()
driver.applications.k70Vsa.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 212 total commands, 21 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.clone()
```

Subgroups

6.1.7.1.1 BitErrorRate

SCPI Commands

```
CALCulate<Window>:BERate
```

class BitErrorRateCls

BitErrorRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(format_py: BerRateFormat, window=Window.Default) → float

```
# SCPI: CALCulate<n>:BERate
value: float = driver.applications.k70Vsa.calculate.bitErrorRate.get(format_py,
↪= enums.BerRateFormat.CURRENT, window = repcap.Window.Default)
```

Queries the Bit Error Rate results. The available results are described in ‘Bit error rate (BER)’. Note that the specified window suffix must refer to a BER result display.

param format_py

Specifies a particular BER result to be queried. If no parameter is specified, the current bit error rate is returned. The parameters for these results are listed in Table ‘Parameters for BER result values’. DSINDEX Queries the index of the identified data sequence found in a known data file. The index starts with 0, that is: the first data sequence in the file is returned as ‘0’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

results: No help available

6.1.7.1.2 Ddem

class DdemCls

Ddem commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.ddem.clone()
```

Subgroups

6.1.7.1.2.1 Burst

class BurstCls

Burst commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.ddem.burst.clone()
```

Subgroups

6.1.7.1.2.2 Length

SCPI Commands

```
CALCulate<Window>:DDEM:BURSt:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → int

```
# SCPI: CALCulate<n>:DDEM:BURSt:LENGth
value: int = driver.applications.k70Vsa.calculate.ddem.burst.length.get(window_
↳ repcap.Window.Default)
```

This command queries the length of a detected burst. Note that since the R&S FSWP VSA application has no knowledge on the ramp length, there is an uncertainty in the burst search algorithm.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

length: integer Number of symbols

6.1.7.1.2.3 Spectrum

class SpectrumCls

Spectrum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.ddem.spectrum.clone()
```

Subgroups

6.1.7.1.2.4 State

SCPI Commands

```
CALCulate<Window>:DDEM:SPECTrum:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DDEM:SPECTrum[:STATe]
value: bool = driver.applications.k70Vsa.calculate.ddem.spectrum.state.
    ↪ get(window = repcap.Window.Default)
```

This command switches the result type transformation to spectrum mode. Spectral evaluation is available for the following result types:

INTRO_CMD_HELP: Prerequisites for this command

- MAGNitude
- PHASe/UPHase
- FREQuency
- Real/Imag (RIMAG)

The result types are defined using the method RsFswp.Applications.K70_Vsa.Calculate.FormatPy.set command (see method RsFswp.Applications.K70_Vsa.Calculate.FormatPy.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:DDEM:SPECTrum[:STATe]
driver.applications.k70Vsa.calculate.ddem.spectrum.state.set(state = False,
↪window = repcap.Window.Default)
```

This command switches the result type transformation to spectrum mode. Spectral evaluation is available for the following result types:

INTRO_CMD_HELP: Prerequisites for this command

- MAGNitude
- PHASe/UPHase
- FREQuency
- Real/Imag (RIMAG)

The result types are defined using the method `RsFswp.Applications.K70_Vsa.Calculate.FormatPy.set` command (see method `RsFswp.Applications.K70_Vsa.Calculate.FormatPy.set`).

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.3 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k70Vsa.calculate.deltaMarker.repcap_deltaMarker_get()
driver.applications.k70Vsa.calculate.deltaMarker.repcap_deltaMarker_set(repcap.
↪DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 16 total commands, 8 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.deltaMarker.clone()
```


Subgroups

6.1.7.1.3.1 Aoff

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:AOff
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:AOff
driver.applications.k70Vsa.calculate.deltaMarker.aoff.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns off all delta markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.1.7.1.3.2 Maximum

class MaximumCls

Maximum commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.deltaMarker.maximum.clone()
```

Subgroups

6.1.7.1.3.3 Apeak

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:APeak
```

class ApeakCls

Apeak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:APeak
driver.applications.k70Vsa.calculate.deltaMarker.maximum.apeak.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command positions the active marker or delta marker on the largest absolute peak value (maximum or minimum) of the selected trace.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.4 Left

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:LEFT
driver.applications.k70Vsa.calculate.deltaMarker.maximum.left.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.5 Next

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:NEXT
driver.applications.k70Vsa.calculate.deltaMarker.maximum.next.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next positive peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.6 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum[:PEAK]
driver.applications.k70Vsa.calculate.deltaMarker.maximum.peak.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.7 Right

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:RIGHT
driver.applications.k70Vsa.calculate.deltaMarker.maximum.right.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value on the trace. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.8 Mburst

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MBURst:START`

class MburstCls

Mburst commands group definition. 1 total commands, 0 Subgroups, 1 group commands

start(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MBURst:START
driver.applications.k70Vsa.calculate.deltaMarker.mburst.start(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves the marker m to the start of the selected result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

```
start_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1)
→ None
```

6.1.7.1.3.9 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.deltaMarker.minimum.clone()
```

Subgroups

6.1.7.1.3.10 Left

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None
```

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:LEFT
driver.applications.k70Vsa.calculate.deltaMarker.minimum.left.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.11 Next

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:NEXT
driver.applications.k70Vsa.calculate.deltaMarker.minimum.next.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.12 Peak

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum[:PEAK]
driver.applications.k70Vsa.calculate.deltaMarker.minimum.peak.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.13 Right

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:RIGHT
driver.applications.k70Vsa.calculate.deltaMarker.minimum.right.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.3.14 State

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTaMarker<m>[:STATe]
value: bool = driver.applications.k70Vsa.calculate.deltaMarker.state.get(window
↳= repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
driver.applications.k70Vsa.calculate.deltaMarker.state.set(state = False,
↪window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.1.7.1.3.15 Trace

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:TRACE
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:TRACE
value: float = driver.applications.k70Vsa.calculate.deltaMarker.trace.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

trace_number: No help available

set(trace_number: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None


```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
driver.applications.k70Vsa.calculate.deltaMarker.trace.set(trace_number = 1.0,
↪window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than ‘Blank’. If necessary, the command activates the marker first.

param trace_number

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.1.7.1.3.16 X

class XCls

X commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.deltaMarker.x.clone()
```

Subgroups

6.1.7.1.3.17 Absolute

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:X:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X:ABSolute
value: float = driver.applications.k70Vsa.calculate.deltaMarker.x.absolute.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command queries the absolute x-value of the selected delta marker in the specified window. The command activates the corresponding delta marker, if necessary.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

result: No help available

6.1.7.1.3.18 Relative

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:X:RELative`

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → int

```
# SCPI: CALCulate<n>:DELTamarker<m>:X:RELative
value: int = driver.applications.k70Vsa.calculate.deltaMarker.x.relative.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command queries the relative position of a delta marker on the x-axis. If necessary, the command activates the delta marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

position: Position of the delta marker in relation to the reference marker.

6.1.7.1.3.19 Y

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:Y`

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:Y
value: float = driver.applications.k70Vsa.calculate.deltaMarker.y.get(window = ↪
↪repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

result: Result at the position of the delta marker. The unit is variable and depends on the one you have currently set. Unit: DBM

6.1.7.1.4 Dlabs**class DlabsCls**

Dlabs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dlabs.clone()
```

Subgroups**6.1.7.1.4.1 State****SCPI Commands**

```
CALCulate<Window>:DLABs:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DLABs:STATE
value: bool = driver.applications.k70Vsa.calculate.dlabs.state.get(window = ↵
↵repcap.Window.Default)
```

Displays an absolute horizontal line in the specified window. This command is only available for eye diagrams.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:DLABs:STATe
driver.applications.k70Vsa.calculate.dlabs.state.set(state = False, window =
↳repcap.Window.Default)
```

Displays an absolute horizontal line in the specified window. This command is only available for eye diagrams.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.4.2 Value

SCPI Commands

```
CALCulate<Window>:DLABs:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:DLABs[:VALue]
value: float = driver.applications.k70Vsa.calculate.dlabs.value.get(window =
↳repcap.Window.Default)
```

Defines value of horizontal absolute line

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

hor_line_abs_pos: Y-value of the absolute horizontal line.

set(hor_line_abs_pos: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:DLABs[:VALue]
driver.applications.k70Vsa.calculate.dlabs.value.set(hor_line_abs_pos = 1.0,
↳window = repcap.Window.Default)
```

Defines value of horizontal absolute line

param hor_line_abs_pos

Y-value of the absolute horizontal line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.5 DRel

class DRelCls

DRel commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dlRel.clone()
```

Subgroups

6.1.7.1.5.1 State

SCPI Commands

```
CALCulate<Window>:DLRel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DLRel:STATe
value: bool = driver.applications.k70Vsa.calculate.dlRel.state.get(window = ↵
↵repcap.Window.Default)
```

Displays a relative horizontal line in the specified window. This command is only available for eye diagrams, and only if an absolute horizontal line is already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.Dlabs.State.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:DLRel:STATe
driver.applications.k70Vsa.calculate.dlRel.state.set(state = False, window = ↵
↵repcap.Window.Default)
```

Displays a relative horizontal line in the specified window. This command is only available for eye diagrams, and only if an absolute horizontal line is already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.Dlabs.State.set) .

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.5.2 Value

SCPI Commands

CALCulate<Window>:DLRel:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:DLRel[:VALue]
value: float = driver.applications.k70Vsa.calculate.dlRel.value.get(window =
↳repcap.Window.Default)
```

Defines or queries the y-value of the relative horizontal line in the specified window. This command is only available for eye diagrams, and only if an absolute horizontal line and a relative horizontal line are already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.Dlabs.State.set and method RsFswp.Applications.K70_Vsa.Calculate.DlRel.State.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

line_rel_pos_rel: Relative distance of the second horizontal line to the first (absolute) horizontal line.

set(*line_rel_pos_rel: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:DLRel[:VALue]
driver.applications.k70Vsa.calculate.dlRel.value.set(line_rel_pos_rel = 1.0,
↳window = repcap.Window.Default)
```

Defines or queries the y-value of the relative horizontal line in the specified window. This command is only available for eye diagrams, and only if an absolute horizontal line and a relative horizontal line are already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.Dlabs.State.set and method RsFswp.Applications.K70_Vsa.Calculate.DlRel.State.set) .

param line_rel_pos_rel

Relative distance of the second horizontal line to the first (absolute) horizontal line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.6 Dsp

class DspCls

Dsp commands group definition. 9 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.clone()
```

Subgroups

6.1.7.1.6.1 Result

class ResultCls

Result commands group definition. 9 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.clone()
```

Subgroups

6.1.7.1.6.2 Capture

class CaptureCls

Capture commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.capture.clone()
```

Subgroups

6.1.7.1.6.3 Bursts

SCPI Commands

```
CALCulate<Window>:DSP:RESult:CAPTure:BURSts
```

class BurstsCls

Bursts commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → int

```
# SCPI: CALCulate<n>:DSP:RESult:CAPTure:BURSts
value: int = driver.applications.k70Vsa.calculate.dsp.result.capture.bursts.
↪get(window = repcap.Window.Default)
```

Queries the number of bursts found across the internal capture buffer. Note that the internal capture buffer is slightly larger than the displayed capture buffer in order to allow for sufficient filter settling times for further processing.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

bursts: integer Number of bursts

6.1.7.1.6.4 Patterns

SCPI Commands

```
CALCulate<Window>:DSP:RESult:CAPTure:PATterns
```

class PatternsCls

Patterns commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → int

```
# SCPI: CALCulate<n>:DSP:RESult:CAPTure:PATterns
value: int = driver.applications.k70Vsa.calculate.dsp.result.capture.patterns.
↪get(window = repcap.Window.Default)
```

Queries the number of patterns found across the internal capture buffer. Note that the internal capture buffer is slightly larger than the displayed capture buffer in order to allow for sufficient filter settling times for further processing.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

patterns: integer Number of patterns

6.1.7.1.6.5 Range

class RangeCls

Range commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.rrange.clone()
```

Subgroups

6.1.7.1.6.6 Current

class CurrentCls

Current commands group definition. 7 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.clone()
```

Subgroups

6.1.7.1.6.7 Burst

class BurstCls

Burst commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.burst.clone()
```

Subgroups

6.1.7.1.6.8 Length

SCPI Commands

```
CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → int`

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:BURSt:LENGth
value: int = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
    ↪burst.length.get(window = repcap.Window.Default)
```

Queries the length of the burst in the current result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

length: Burst length in samples Unit: none

6.1.7.1.6.9 Present

SCPI Commands

CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:PRESent

class PresentCls

Present commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:BURSt:PRESent
value: bool = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
↳burst.present.get(window = repcap.Window.Default)
```

Queries whether a burst is present or not in the current result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

present: ON | OFF | 0 | 1 OFF | 0 Burst not available. ON | 1 Burst available

6.1.7.1.6.10 Start

SCPI Commands

CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:STARt

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → int

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:BURSt:STARt
value: int = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
↳burst.start.get(window = repcap.Window.Default)
```

Queries the burst start in the current result range as an offset to the capture buffer start. Tip: To determine the capture buffer start, use the method **RsFswp.Applications.K70_Vsa.Display.Window.Trace.X.Scale.Start.get_** command for a window with a capture buffer display.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

start: Offset in symbols from the capture buffer start. Unit: sym

6.1.7.1.6.11 Pattern**class PatternCls**

Pattern commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.pattern.clone()
```

Subgroups**6.1.7.1.6.12 Confidence****SCPI Commands**

```
CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:CONFidence
```

class ConfidenceCls

Confidence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → int

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:PATtern:CONFidence
value: int = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
pattern.confidence.get(window = repcap.Window.Default)
```

Queries the confidence with which the pattern was detected in the current result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

confidence: Percentage of correct identification of pattern Range: 0 to 100, Unit: percent

6.1.7.1.6.13 Correct

SCPI Commands

CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:CORRect

class CorrectCls

Correct commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:PATtern:CORRect
value: bool = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
↳ pattern.correct.get(window = repcap.Window.Default)
```

Queries whether the pattern is correct or not in the current result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

correct: ON | OFF | 0 | 1 OFF | 0 Pattern not correct. ON | 1 Pattern correct

6.1.7.1.6.14 Present

SCPI Commands

CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:PRESent

class PresentCls

Present commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:PATtern:PRESent
value: bool = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
↳ pattern.present.get(window = repcap.Window.Default)
```

Queries whether a pattern is present or not in the current result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

present: ON | OFF | 0 | 1 OFF | 0 Pattern not available. ON | 1 Pattern available

6.1.7.1.6.15 Start

SCPI Commands

```
CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → int

```
# SCPI: CALCulate<n>:DSP:RESult:RRANge:CURRent:PATtern:STARt
value: int = driver.applications.k70Vsa.calculate.dsp.result.rrange.current.
        ↪ pattern.start.get(window = repcap.Window.Default)
```

Queries the pattern start in the current result range as an offset to the capture buffer start. **Tip:** To determine the capture buffer start, use the method **RsFswp.Applications.K70_Vsa.Display.Window.Trace.X.Scale.Start.get_** command for a window with a capture buffer display.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

start: Offset in symbols from the capture buffer start. Unit: sym

6.1.7.1.7 Elin

class ElinCls

Elin commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.elin.clone()
```

Subgroups

6.1.7.1.7.1 State

SCPI Commands

```
CALCulate<Window>:ELIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:ELIN:STATe
value: bool = driver.applications.k70Vsa.calculate.elin.state.get(window = ↵
↵repcap.Window.Default)
```

This command restricts the evaluation range. The evaluation range is considered for the following display types:

INTRO_CMD_HELP: Prerequisites for this command

- eye diagrams
- constellation diagrams
- modulation accuracy
- statistic displays
- spectrum displays

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | 1 The evaluation range extends from the start value defined by CALC:ELIN1:VAL to the stop value defined by CALC:ELIN2:VAL (see method RsFswp.Applications.K70_Vsa.Calculate.Elin.Value.set) . OFF | 0 The complete result area is evaluated.

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:ELIN:STATe
driver.applications.k70Vsa.calculate.elin.state.set(state = False, window = ↵
↵repcap.Window.Default)
```

This command restricts the evaluation range. The evaluation range is considered for the following display types:

INTRO_CMD_HELP: Prerequisites for this command

- eye diagrams
- constellation diagrams
- modulation accuracy
- statistic displays
- spectrum displays

param state

ON | 1 The evaluation range extends from the start value defined by CALC:ELIN1:VAL to the stop value defined by CALC:ELIN2:VAL (see method RsFswp.Applications.K70_Vsa.Calculate.Elin.Value.set) . OFF | 0 The complete result area is evaluated.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.7.2 Value

SCPI Commands

```
CALCulate<Window>:ELIN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:ELIN[:VALue]
value: float = driver.applications.k70Vsa.calculate.elin.value.get(window = repcap.Window.Default)
```

Defines the start and stop values for the evaluation range (see method RsFswp.Applications.K70_Vsa.Calculate.Elin.State.set).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

left_disp: Range: 0 to 1000000, Unit: SYM

set(left_disp: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:ELIN[:VALue]
driver.applications.k70Vsa.calculate.elin.value.set(left_disp = 1.0, window = repcap.Window.Default)
```

Defines the start and stop values for the evaluation range (see method RsFswp.Applications.K70_Vsa.Calculate.Elin.State.set).

param left_disp

Range: 0 to 1000000, Unit: SYM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.8 Feed

SCPI Commands

```
CALCulate<Window>:FEED
```

class FeedCls

Feed commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:FEED
value: str = driver.applications.k70Vsa.calculate.feed.get(window = repcap.Window.Default)
```

Selects the signal source (and for the equalizer also the result type) for evaluation. Note that this command is maintained for compatibility reasons only. Use the LAYout commands for new remote control programs (see ‘Working with windows in the display’) . Only for the ‘Equalizer Impulse Response’ and ‘Equalizer Frequency Response’, as well as the multi-source diagrams, this command is required.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

feed: string ‘XTIM:DDEM:MEAS’ Measured signal ‘XTIM:DDEM:REF’ Reference signal ‘XTIM:DDEM:ERR:VECT’ Error vector ‘XTIM:DDEM:ERR:MPH’ Modulation errors ‘XTIM:DDEM:MACC’ Modulation accuracy ‘XTIM:DDEM:SYMB’ Symbol table ‘TCAP’ Capture buffer ‘XTIM:DDEM:IMP’ Equalizer Impulse Response ‘XFR:DDEM:RAT’ Equalizer Frequency Response ‘XFR:DDEM:IRAT’ Equalizer Channel Frequency Response Group Delay XTIM:DDEM:TCAP:ERR Spectrum of Real/Imag for capture buffer and error vector XTIM:DDEM:MEAS:ERR Spectrum of Real/Imag for measurement and error vector

set(feed: str, window=Window.Default) → None

```
# SCPI: CALCulate<n>:FEED
driver.applications.k70Vsa.calculate.feed.set(feed = '1', window = repcap.
↳Window.Default)
```

Selects the signal source (and for the equalizer also the result type) for evaluation. Note that this command is maintained for compatibility reasons only. Use the LAYout commands for new remote control programs (see ‘Working with windows in the display’) . Only for the ‘Equalizer Impulse Response’ and ‘Equalizer Frequency Response’, as well as the multi-source diagrams, this command is required.

param feed

string ‘XTIM:DDEM:MEAS’ Measured signal ‘XTIM:DDEM:REF’ Reference signal ‘XTIM:DDEM:ERR:VECT’ Error vector ‘XTIM:DDEM:ERR:MPH’ Modulation errors ‘XTIM:DDEM:MACC’ Modulation accuracy ‘XTIM:DDEM:SYMB’ Symbol table ‘TCAP’ Capture buffer ‘XTIM:DDEM:IMP’ Equalizer Impulse Response ‘XFR:DDEM:RAT’ Equalizer Frequency Response ‘XFR:DDEM:IRAT’ Equalizer Channel Frequency Response Group Delay XTIM:DDEM:TCAP:ERR Spectrum of Real/Imag for capture buffer and error vector XTIM:DDEM:MEAS:ERR Spectrum of Real/Imag for measurement and error vector

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.9 FormatPy

SCPI Commands

```
CALCulate<Window>:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → TraceFormat


```
# SCPI: CALCulate<n>:FORMat
value: enums.TraceFormat = driver.applications.k70Vsa.calculate.formatPy.
↳get(window = repcap.Window.Default)
```

This command defines the result type of the traces. Which parameters are available depends on the setting for the data source (see method [RsFswp.Layout.Add.Window.get](#) and Table ‘Available result types depending on data source’). Whether the result type shows absolute or relative values is defined using the DISP:WIND:TRAC:Y:MODE command (see method [RsFswp.Display.Window.Subwindow.Trace.Y.Scale.Mode.set](#)).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

format_py: MAGNitude | PHASe | UPHase | RIMag | FREQuency | COMP | CONS | IEYE | QEYE | FEYE | CONF | COVF | RCONstellation | RSUMmary | BERate | GDElay | MOVerview | BIN | OCT | DEC | HEX | NONE MAGNitude Magnitude Absolute MOVerview Magnitude Overview Absolute (entire capture buffer) PHASe ‘Phase Wrap’ UPHase ‘Phase Unwrap’ RIMag ‘Real/Imag (I/Q)’ FREQuency ‘Frequency Absolute’ COMP ‘Vector I/Q’ CONS ‘Constellation I/Q’ IEYE ‘Eye Diagram Real (I)’ QEYE ‘Eye Diagram Imag (Q)’ FEYE ‘Eye Diagram Frequency’ CONF ‘Constellation Frequency’ COVF ‘Vector Frequency’ RCONstellation ‘Constellation I/Q (Rotated)’ RSUMmary ‘Result summary’ BERate ‘Bit error rate’ GDElay ‘Frequency Response Group Delay’ BIN ‘Symbol table’ in binary format OCT ‘Symbol table’ in octal format DEC ‘Symbol table’ in decimal format HEX ‘Symbol table’ in hexadecimal format

set(format_py: TraceFormat, window=Window.Default) → None

```
# SCPI: CALCulate<n>:FORMat
driver.applications.k70Vsa.calculate.formatPy.set(format_py = enums.TraceFormat.
↳BERate, window = repcap.Window.Default)
```

This command defines the result type of the traces. Which parameters are available depends on the setting for the data source (see method [RsFswp.Layout.Add.Window.get](#) and Table ‘Available result types depending on data source’). Whether the result type shows absolute or relative values is defined using the DISP:WIND:TRAC:Y:MODE command (see method [RsFswp.Display.Window.Subwindow.Trace.Y.Scale.Mode.set](#)).

param format_py

MAGNitude | PHASe | UPHase | RIMag | FREQuency | COMP | CONS | IEYE | QEYE | FEYE | CONF | COVF | RCONstellation | RSUMmary | BERate | GDElay | MOVerview | BIN | OCT | DEC | HEX | NONE MAGNitude Magnitude Absolute MOVerview Magnitude Overview Absolute (entire capture buffer) PHASe ‘Phase Wrap’ UPHase ‘Phase Unwrap’ RIMag ‘Real/Imag (I/Q)’ FREQuency ‘Frequency Absolute’ COMP ‘Vector I/Q’ CONS ‘Constellation I/Q’ IEYE ‘Eye Diagram Real (I)’ QEYE ‘Eye Diagram Imag (Q)’ FEYE ‘Eye Diagram Frequency’ CONF ‘Constellation Frequency’ COVF ‘Vector Frequency’ RCONstellation ‘Constellation I/Q (Rotated)’ RSUMmary ‘Result summary’ BERate ‘Bit error rate’ GDElay ‘Frequency Response Group Delay’ BIN ‘Symbol table’ in binary format OCT ‘Symbol table’ in octal format DEC ‘Symbol table’ in decimal format HEX ‘Symbol table’ in hexadecimal format

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.10 Fsk

class FskCls

Fsk commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.fsk.clone()
```

Subgroups

6.1.7.1.10.1 Deviation

class DeviationCls

Deviation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.fsk.deviation.clone()
```

Subgroups

6.1.7.1.10.2 Compensation

SCPI Commands

```
CALCulate<Window>:FSK:DEViation:COMPensation
```

class CompensationCls

Compensation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:FSK:DEViation:COMPensation
value: bool = driver.applications.k70Vsa.calculate.fsk.deviation.compensation.
    ↪ get(window = repcap.Window.Default)
```

This command defines whether the deviation error is compensated for when calculating the frequency error for FSK modulation. Note that this command is maintained for compatibility reasons only. For newer remote programs, use [SENSe:]DDEMod:NORMalize:FDERror.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | 1 Scales the reference signal to the actual deviation of the measurement signal. OFF | 0 Uses the entered nominal deviation for the reference signal.

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:FSK:DEViation:COMPensation
driver.applications.k70Vsa.calculate.fsk.deviation.compensation.set(state = False, window = repcap.Window.Default)
```

This command defines whether the deviation error is compensated for when calculating the frequency error for FSK modulation. Note that this command is maintained for compatibility reasons only. For newer remote programs, use [SENSe:]DDEMod:NORMalize:FDERror.

param state

ON | 1 Scales the reference signal to the actual deviation of the measurement signal.
OFF | 0 Uses the entered nominal deviation for the reference signal.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.10.3 Reference**class ReferenceCls**

Reference commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.fsk.deviation.reference.clone()
```

Subgroups**6.1.7.1.10.4 Relative****SCPI Commands**

```
CALCulate<Window>:FSK:DEViation:REference:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:FSK:DEViation:REference:RELative
value: float = driver.applications.k70Vsa.calculate.fsk.deviation.reference.  
relative.get(window = repcap.Window.Default)
```

This command defines the deviation to the reference frequency for FSK modulation as a multiple of the symbol rate.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

fsk_ref_dev: Range: 0.1 to 60, Unit: none

set(fsk_ref_dev: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:FSK:DEVIation:REference:RELative
driver.applications.k70Vsa.calculate.fsk.deviation.reference.relative.set(fsk_
↪ref_dev = 1.0, window = repcap.Window.Default)
```

This command defines the deviation to the reference frequency for FSK modulation as a multiple of the symbol rate.

param fsk_ref_dev

Range: 0.1 to 60, Unit: none

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.10.5 Value

SCPI Commands

CALCulate<Window>:FSK:DEVIation:REference:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:FSK:DEVIation:REference[:VALue]
value: float = driver.applications.k70Vsa.calculate.fsk.deviation.reference.
↪value.get(window = repcap.Window.Default)
```

This command defines the deviation to the reference frequency for FSK modulation as an absolute value in Hz.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

fsk_ref_dev_abs_res: Range: 10.0 to 256e9, Unit: HZ

set(fsk_ref_dev_abs_res: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:FSK:DEVIation:REference[:VALue]
driver.applications.k70Vsa.calculate.fsk.deviation.reference.value.set(fsk_ref_
↪dev_abs_res = 1.0, window = repcap.Window.Default)
```

This command defines the deviation to the reference frequency for FSK modulation as an absolute value in Hz.

param fsk_ref_dev_abs_res

Range: 10.0 to 256e9, Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11 Limit

class LimitCls

Limit commands group definition. 110 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.clone()
```

Subgroups

6.1.7.1.11.1 Maccuracy

class MaccuracyCls

Maccuracy commands group definition. 110 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.clone()
```

Subgroups

6.1.7.1.11.2 CfError

class CfErrorCls

CfError commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.clone()
```

Subgroups

6.1.7.1.11.3 Current

class CurrentCls

Current commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.current.clone()
```

Subgroups

6.1.7.1.11.4 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:CURRent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.
    ↪current.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.5 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:CURRent:STATE
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.
↳current.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:CURRent:STATE
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.current.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.6 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:VALUE
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:CURRent:VALUE
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.
↳current.value.get(window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:CURRent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.current.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.7 Mean

class MeanCls

Mean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.mean.clone()
```

Subgroups

6.1.7.1.11.8 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:MEAN:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:MEAN[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.mean.
↪result.get(window = repcap.Window.Default)
```


This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.9 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:MEAN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:MEAN:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.mean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:MEAN:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.mean.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.10 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:CFERror:MEAN:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:MEAN:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.
↳mean.value.get(window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:MEAN:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.mean.value.
↳set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.11 Peak

class PeakCls

Peak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.peak.clone()
```

Subgroups

6.1.7.1.11.12 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:PEAK:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:PEAK[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.peak.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.13 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:PEAK:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:PEAK:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.peak.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:PEAK:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.peak.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.14 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:CFERror:PEAK:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:PEAK:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.
↪peak.value.get(window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:CFERror:PEAK:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.cfError.peak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the limit for the current, peak or mean center frequency error limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 1000000, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.15 Default**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:DEFault
driver.applications.k70Vsa.calculate.limit.maccuracy.default.set(window =
↳repcap.Window.Default)
```

Restores the default limits and deactivates all checks in all windows.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

set_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

6.1.7.1.11.16 Evm**class EvmCls**

Evm commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.clone()
```

Subgroups**6.1.7.1.11.17 Pcurrent****class PcurrentCls**

Pcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pcurrent.clone()
```

Subgroups

6.1.7.1.11.18 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pcurrent.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.19 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PCURrent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pcurrent.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PCURrent:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pcurrent.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.20 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.
↪pcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return
limit_value: Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pcurrent.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value
Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.21 Pmean**class PmeanCls**

Pmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.clone()
```

Subgroups**6.1.7.1.11.22 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.23 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.state.set(state,
↳ False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.24 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.
↳ value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.pmean.value.set(limit_
↪value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.25 Ppeak

class PpeakCls

Ppeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak.clone()
```

Subgroups

6.1.7.1.11.26 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.27 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PPEak:STATe value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak. ↪ state.get(window = repcap.Window.Default)</pre>

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PPEak:STATe driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak.state.set(state, ↪ False, window = repcap.Window.Default)</pre>
--

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.28 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak.
↳ value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: 0.0 to 100, Unit: %

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:PPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.ppeak.value.set(limit_
↳ value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.29 Rcurrent

class RcurrentCls

Rcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rcurrent.clone()
```

Subgroups

6.1.7.1.11.30 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rcurrent.
    ↪ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.31 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RCURrent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rcurrent.
    ↪ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RCURrent:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rcurrent.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.32 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.
↪rcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rcurrent.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.33 Rmean**class RmeanCls**

Rmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.clone()
```

Subgroups**6.1.7.1.11.34 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.
    ↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.35 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.state.set(state,
↳ False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.36 Value

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.
↳ value.get(window = repcap.Window.Default)
```


This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rmean.value.set(limit_
↪value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.37 Rpeak

class RpeakCls

Rpeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.clone()
```

Subgroups

6.1.7.1.11.38 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.39 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.state.set(state,
↳ False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.40 Value

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.
↪value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:EVM:RPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.evm.rpeak.value.set(limit_
↪value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean EVM (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.41 FdError

class FdErrorCls

FdError commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.clone()
```

Subgroups

6.1.7.1.11.42 Current

class CurrentCls

Current commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.current.clone()
```

Subgroups

6.1.7.1.11.43 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:CURRent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:CURRent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.
    ↪current.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.44 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:CURRent:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:CURRent:STATE
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.
↳current.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:CURRent:STATE
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.current.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.45 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:CURRent:VALUE
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:CURRent:VALUE
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.
↳current.value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: 0.0 to 1000000

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:CURRent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.current.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

Range: 0.0 to 1000000

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.46 Mean

class MeanCls

Mean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.mean.clone()
```

Subgroups

6.1.7.1.11.47 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:MEAN:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:MEAN[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.mean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.48 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDERror:MEAN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:MEAN:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.mean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:MEAN:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.mean.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.49 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:FDERror:MEAN:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:MEAN:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.
↳mean.value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: 0.0 to 1000000

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:MEAN:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.mean.value.
↳set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

Range: 0.0 to 1000000

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.50 Peak

class PeakCls

Peak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.peak.clone()
```

Subgroups

6.1.7.1.11.51 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:PEAK:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:PEAK[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.peak.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.52 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDError:PEAK:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDError:PEAK:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.peak.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:PEAK:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.peak.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.53 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FDERror:PEAK:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:PEAK:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.
↪peak.value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: 0.0 to 1000000

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FDERror:PEAK:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.fdError.peak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean center frequency deviation error. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

Range: 0.0 to 1000000

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.54 FreqError**class FreqErrorCls**

FreqError commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.clone()
```

Subgroups**6.1.7.1.11.55 Pcurrent****class PcurrentCls**

Pcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pcurrent.clone()
```

Subgroups**6.1.7.1.11.56 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    pcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.57 State

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:FERRor:PCURrent:STATE`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PCURrent:STATE
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↳ pcurrent.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PCURrent:STATE
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pcurrent.state.
    ↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.58 Value

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:FERRor:PCURrent:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↳ pcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pcurrent.value.
    ↳ set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.59 Pmean

class PmeanCls

Pmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pmean.clone()
```

Subgroups

6.1.7.1.11.60 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪ pmean.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.61 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪ pmean.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pmean.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.62 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↪pmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return
limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.pmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value
the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.63 Ppeak**class PpeakCls**

Ppeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.ppeak.clone()
```

Subgroups**6.1.7.1.11.64 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪ppeak.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.65 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PPEak:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↳ ppeak.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.ppeak.state.
    ↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.66 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:PPEak:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↳ ppeak.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:PPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.ppeak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.67 Rcurrent

class RcurrentCls

Rcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rcurrent.clone()
```

Subgroups

6.1.7.1.11.68 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↳ rcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.69 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RCURrent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RCURrent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↳ rcurrent.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RCURrent:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rcurrent.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.70 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:FERRor:RCURrent:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↳ rcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rcurrent.value.
    ↳ set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.71 Rmean

class RmeanCls

Rmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rmean.clone()
```

Subgroups

6.1.7.1.11.72 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪ rmean.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.73 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪ rmean.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rmean.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.74 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↪rmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.75 Rpeak**class RpeakCls**

Rpeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rpeak.clone()
```

Subgroups**6.1.7.1.11.76 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:FERRor:RPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
    ↪rpeak.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.77 State

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:FERRor:RPEak:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↳ rpeak.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rpeak.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.78 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:FERRor:RPEak:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.
↳ rpeak.value.get(window = repcap.Window.Default)
```


This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:FERRor:RPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.freqError.rpeak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean frequency error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical. This command is available for FSK modulation only.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: Hz

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.79 Merror

class MerrorCls

Merror commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.clone()
```

Subgroups

6.1.7.1.11.80 Pcurrent

class PcurrentCls

Pcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pcurrent.clone()
```

Subgroups

6.1.7.1.11.81 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
    ↪ pcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.82 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PCURrent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PCURrent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
    ↪ pcurrent.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PCURrent:STATE
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pcurrent.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.83 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PCURrent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
↪pcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return
limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pcurrent.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value
the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.84 Pmean**class PmeanCls**

Pmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pmean.clone()
```

Subgroups**6.1.7.1.11.85 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pmean.
    ↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see 'Result summary'.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

result: No help available

6.1.7.1.11.86 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pmean.
↳state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pmean.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.87 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
↳pmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.pmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.88 Ppeak

class PpeakCls

Ppeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.ppeak.clone()
```

Subgroups

6.1.7.1.11.89 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.ppeak.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.90 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.ppeak.
↳state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.ppeak.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.91 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
      ↳ppeak.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:PPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.ppeak.value.
      ↳set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.92 Rcurrent

class RcurrentCls

Rcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rcurrent.clone()
```

Subgroups

6.1.7.1.11.93 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
    ↪rcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.94 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RCURrent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
    ↪rcurrent.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RCURrent:STATE
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rcurrent.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.95 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
↪rcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rcurrent.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.96 Rmean**class RmeanCls**

Rmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rmean.clone()
```

Subgroups**6.1.7.1.11.97 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rmean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see 'Result summary'.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

result: No help available

6.1.7.1.11.98 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rmean.
↳state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rmean.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.99 Value

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
↳rmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.100 Rpeak

class RpeakCls

Rpeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rpeak.clone()
```

Subgroups

6.1.7.1.11.101 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rpeak.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.102 State

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rpeak.
↳state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rpeak.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.103 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.merror.
    ↳ rpeak.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:MERRor:RPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.merror.rpeak.value.
    ↳ set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean magnitude error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 100, Unit: %

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.104 Offset

class OffsetCls

Offset commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.clone()
```

Subgroups

6.1.7.1.11.105 Current

class CurrentCls

Current commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.current.clone()
```

Subgroups

6.1.7.1.11.106 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOFfset:CURRent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOFfset:CURRent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.
    ↪current.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.107 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:CURRent:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:CURRent:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.
↳ current.state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:CURRent:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.current.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.108 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:CURRent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:CURRent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.
↳ current.value.get(window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: -200.0 to 0.0, Unit: DB

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:CURRent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.current.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: -200.0 to 0.0, Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.109 Mean

class MeanCls

Mean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.mean.clone()
```

Subgroups

6.1.7.1.11.110 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:MEAN:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:MEAN[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.mean.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.111 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:MEAN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:MEAN:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.mean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:MEAN:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.mean.state.
↳ set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.112 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:OOffset:MEAN:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:MEAN:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.
↳mean.value.get(window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: -200.0 to 0.0, Unit: DB

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:MEAN:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.mean.value.
↳set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: -200.0 to 0.0, Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.113 Peak

class PeakCls

Peak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.peak.clone()
```

Subgroups

6.1.7.1.11.114 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:PEAK:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:PEAK[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.peak.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.115 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:PEAK:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:PEAK:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.peak.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:PEAK:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.peak.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.116 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:OOffset:PEAK:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:PEAK:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.
↪peak.value.get(window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: -200.0 to 0.0, Unit: DB

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:OOffset:PEAK:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.ooffset.peak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the upper limit for the current, peak or mean I/Q offset. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: -200.0 to 0.0, Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.117 Perror**class PerrorCls**

Perror commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.clone()
```

Subgroups**6.1.7.1.11.118 Pcurrent****class PcurrentCls**

Pcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pcurrent.clone()
```

Subgroups**6.1.7.1.11.119 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
    pcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.120 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:PERRor:PCURrent:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PCURrent:STATE value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror. ↳ pcurrent.state.get(window = repcap.Window.Default)</pre>

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PCURrent:STATE driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pcurrent.state. ↳ set(state = False, window = repcap.Window.Default)</pre>
--

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.121 Value

SCPI Commands

```
CALCulate<Window>:LIMIT:MACCuracy:PERRor:PCURrent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMIT:MACCuracy:PERRor:PCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
    ↳ pcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMIT:MACCuracy:PERRor:PCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pcurrent.value.
    ↳ set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.122 Pmean

class PmeanCls

Pmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pmean.clone()
```

Subgroups

6.1.7.1.11.123 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pmean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.124 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pmean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pmean.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.125 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
↪pmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.pmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.126 Ppeak**class PpeakCls**

Ppeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.ppeak.clone()
```

Subgroups**6.1.7.1.11.127 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.ppeak.
    ↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.128 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PPEak:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.ppeak.
↳state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.ppeak.state.
↳set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.129 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:PPEak:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
↳ppeak.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:PPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.ppeak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.130 Rcurrent

class RcurrentCls

Rcurrent commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rcurrent.clone()
```

Subgroups

6.1.7.1.11.131 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:RCURrent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RCURrent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
↪rcurrent.result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.132 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:PERRor:RCURrent:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RCURrent:STATe value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror. ↳ rcurrent.state.get(window = repcap.Window.Default)</pre>

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RCURrent:STATe driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rcurrent.state. ↳ set(state = False, window = repcap.Window.Default)</pre>
--

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.133 Value

SCPI Commands

`CALCulate<Window>:LIMit:MACCuracy:PERRor:RCURrent:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RCURrent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
    ↳ rcurrent.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RCURrent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rcurrent.value.
    ↳ set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.134 Rmean

class RmeanCls

Rmean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rmean.clone()
```

Subgroups

6.1.7.1.11.135 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:RMEan:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RMEan[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rmean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.136 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:RMEan:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RMEan:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rmean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RMEan:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rmean.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.137 Value**SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:RMEan:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RMEan:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
↪rmean.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RMEan:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rmean.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.138 Rpeak**class RpeakCls**

Rpeak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rpeak.clone()
```

Subgroups**6.1.7.1.11.139 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:PERRor:RPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RPEak[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rpeak.
    ↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see 'Result summary'.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

result: No help available

6.1.7.1.11.140 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:PERRor:RPEak:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RPEak:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rpeak.
    state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RPEak:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rpeak.state.
    set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.141 Value

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:PERRor:RPEak:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RPEak:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.perror.
    rpeak.value.get(window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:PERRor:RPEak:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.perror.rpeak.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the value for the current, peak or mean phase error (peak or RMS) limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

the value x (x0) defines the interval [-x; x] Range: 0.0 to 360, Unit: deg

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.142 Rho

class RhoCls

Rho commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.clone()
```

Subgroups

6.1.7.1.11.143 Current

class CurrentCls

Current commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.clone()
```

Subgroups

6.1.7.1.11.144 Result

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:CURRent:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:CURRent[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.145 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:CURRent:STAtE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:CURRent:STAtE
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:CURRent:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.146 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:CURRent:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:CURRent:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.
↪value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return
limit_value: Range: 0.0 to 1.0, Unit: none

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:CURRent:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.rho.current.value.
↪set(limit_value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value
Range: 0.0 to 1.0, Unit: none

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.147 Mean**class MeanCls**

Mean commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.clone()
```

Subgroups**6.1.7.1.11.148 Result****SCPI Commands**

```
CALCulate<Window>:LIMit:MACCuracy:RHO:MEAN:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:MEAN[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.
↳ result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.149 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:MEAN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:MEAN:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.
↳ state.get(window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:MEAN:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.state.set(state =
↳ False, window = repcap.Window.Default)
```

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.150 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:MEAN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:MEAN:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.
↳ value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

limit_value: Range: 0.0 to 1.0, Unit: none

set(limit_value: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMIT:MACCuracy:RHO:MEAN:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.rho.mean.value.set(limit_
↪value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 1.0, Unit: none

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.151 Peak

class PeakCls

Peak commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak.clone()
```

Subgroups

6.1.7.1.11.152 Result

SCPI Commands

```
CALCulate<Window>:LIMIT:MACCuracy:RHO:PEAK:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:LIMIT:MACCuracy:RHO:PEAK[:RESult]
value: str = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak.
↪result.get(window = repcap.Window.Default)
```

This command queries whether the limit for the specified result type and limit type was violated. For details on result types and limit types see ‘Result summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

result: No help available

6.1.7.1.11.153 State

SCPI Commands

CALCulate<Window>:LIMit:MACCuracy:RHO:PEAK:STATe
--

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:PEAK:STATe value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak. ↳state.get(window = repcap.Window.Default)</pre>
--

This command switches the limit check for the selected result type and limit type on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:PEAK:STATe driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak.state.set(state = ↳False, window = repcap.Window.Default)</pre>
--

This command switches the limit check for the selected result type and limit type on or off.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.11.154 Value

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:RHO:PEAK:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:PEAK:VALue
value: float = driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak.
↳ value.get(window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

limit_value: Range: 0.0 to 1.0, Unit: none

set(*limit_value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:RHO:PEAK:VALue
driver.applications.k70Vsa.calculate.limit.maccuracy.rho.peak.value.set(limit_
↳ value = 1.0, window = repcap.Window.Default)
```

This command defines the lower limit for the current, peak or mean Rho limit. Note that the limits for the current and the peak value are always kept identical.

param limit_value

Range: 0.0 to 1.0, Unit: none

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.11.155 State

SCPI Commands

```
CALCulate<Window>:LIMit:MACCuracy:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:STATe
value: bool = driver.applications.k70Vsa.calculate.limit.maccuracy.state.
↳ get(window = repcap.Window.Default)
```

Limits checks for all evaluations based on modulation accuracy (e.g. 'Result Summary') are enabled or disabled.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:LIMit:MACCuracy:STATe
driver.applications.k70Vsa.calculate.limit.maccuracy.state.set(state = False,
↪ window = repcap.Window.Default)
```

Limits checks for all evaluations based on modulation accuracy (e.g. 'Result Summary') are enabled or disabled.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.12 Marker<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.calculate.marker.repcap_marker_get()
driver.applications.k70Vsa.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 39 total commands, 11 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.clone()
```

Subgroups

6.1.7.1.12.1 Aoff

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:AOFF`

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:AOFF
driver.applications.k70Vsa.calculate.marker.aoff.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

This command turns off all markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.7.1.12.2 Function

class FunctionCls

Function commands group definition. 20 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.function.clone()
```

Subgroups

6.1.7.1.12.3 Ddemod

class DdemodCls

Ddemod commands group definition. 20 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.function.ddemod.clone()
```

Subgroups

6.1.7.1.12.4 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:DDEMod:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(result_type: DdemResultType, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:DDEMod:RESult
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ result.get(result_type = enums.DdemResultType.ADR, window = repcap.Window.
↳ Default, marker = repcap.Marker.Default)
```

No command help available

param result_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.5 Statistic

class StatisticCls

Statistic commands group definition. 19 total commands, 16 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.function.ddemod.statistic.clone()
```

Subgroups

6.1.7.1.12.6 Adroop

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:ADRoop
```

class AdroopCls

Adroop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type*: *ResultTypeStat*, *window*=*Window.Default*, *marker*=*Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:ADRoop
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳statistic.adroop.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the amplitude droop error measurement performed for digital demodulation. The output values are the same as those provided in the ‘Modulation Accuracy’ table (see ‘Result summary’).

param result_type

none Amplitude droop in dB/symbol (for current sweep) AVG Amplitude droop in dB/symbol, evaluating the linear average value over several sweeps RPE Peak value for amplitude droop over several sweeps SDEV Standard deviation of amplitude droop PCTL 95 percentile value of amplitude droop

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

result: No help available

6.1.7.1.12.7 All

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → str

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:ALL
value: str = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.all.get(window = repcap.Window.Default, marker = repcap.Marker.
↳ Default)
```

The command queries all results of the ‘Result Summary’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

results: Comma-separated list of result values in the order described below. Note the last rows differ from the Result Summary display.

6.1.7.1.12.8 CfError

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:CFError
```

class CfErrorCls

CfError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:CFError
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.cfError.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the carrier frequency error measurement performed for digital demodulation. The output values are the same as those provided in the ‘Modulation Accuracy’ table.

param result_type

none Carrier frequency error for current sweep
AVG Average carrier frequency error over several sweeps
RPE Peak carrier frequency error over several sweeps
SDEV Standard deviation of frequency error
PCTL 95 percentile value of frequency error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

result: No help available

6.1.7.1.12.9 Evm

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:EVM
--

class EvmCls

Evm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type*: *ResultTypeStat*, *window*=*Window.Default*, *marker*=*Marker.Default*) → float

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:EVM value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod. ↳ statistic.evm.get(result_type = enums.ResultTypeStat.AVG, window = repcap. ↳ Window.Default, marker = repcap.Marker.Default)</pre>
--

This command queries the results of the error vector magnitude measurement of digital demodulation. The output values are the same as those provided in the ‘Modulation Accuracy’ table .

param result_type

none RMS EVM value of display points of current sweep AVG Average of RMS EVM values over several sweeps PAVG Average of maximum EVM values over several sweeps PCTL 95% percentile of RMS EVM value over several sweeps PEAK Maximum EVM over all symbols of current sweep PPCT 95% percentile of maximum EVM values over several sweeps PSD Standard deviation of maximum EVM values over several sweeps RPE Maximum value of RMS EVM over several sweeps SDEV Standard deviation of EVM values over several sweeps TPE Maximum EVM over all display points over several sweeps

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

result: No help available

6.1.7.1.12.10 FdError

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:FDError
```

class FdErrorCls

FdError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:FDError
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.fdError.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
    ↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the FSK deviation error of FSK modulated signals.

param result_type

none Deviation error for current sweep. AVG Average FSK deviation error. RPE Peak FSK deviation error. SDEV Standard deviation of FSK deviation error. PCTL 95 percentile value of FSK deviation error.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.11 Fsk

class FskCls

Fsk commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.function.ddemod.statistic.fsk.
    ↳ clone()
```

Subgroups

6.1.7.1.12.12 Cfdrift

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:FSK:CFDRift
```

class CfdriftCls

Cfdrift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:FSK:CFDRift
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.fsk.cfdrift.get(result_type = enums.ResultTypeStat.AVG, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the carrier frequency drift for FSK modulated signals.

param result_type

none Carrier frequency drift for current sweep. AVG Average FSK carrier frequency drift over several sweeps. RPE Peak FSK carrier frequency drift over several sweeps. SDEV Standard deviation of FSK carrier frequency drift. PCTL 95 percentile value of FSK carrier frequency drift.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.13 Derror

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:FSK:DERRor
```

class DerrorCls

Derror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:FSK:DERRor
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.fsk.derror.get(result_type = enums.ResultTypeStat.AVG, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the frequency error of FSK modulated signals.

param result_type

none RMS frequency error of display points of current sweep
 AVG Average of frequency errors over several sweeps
 PAVG Average of maximum frequency errors over several sweeps
 PCTL 95% percentile of frequency error over several sweeps
 PEAK Maximum frequency error over all symbols of current sweep
 PPCT 95% percentile of maximum frequency errors over several sweeps
 PSD Standard deviation of maximum frequency errors over several sweeps
 RPE Maximum value of frequency error over several sweeps
 SDEV Standard deviation of frequency errors over several sweeps
 TPE Maximum frequency error over all display points over several sweeps

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.14 Mdeviation**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDemod:STATistic:FSK:MDEVIation
```

class MdeviationCls

Mdeviation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDemod:STATistic:FSK:MDEVIation
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.fsk.mdeviation.get(result_type = enums.ResultTypeStat.AVG, window =
    ↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the measurement deviation of FSK modulated signals.

param result_type

none Measurement deviation for current sweep. AVG Average FSK measurement deviation over several sweeps. RPE Peak FSK measurement deviation over several sweeps. SDEV Standard deviation of FSK measurement deviation. PCTL 95 percentile value of FSK measurement deviation.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.15 Rdeviation

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:FSK:RDEViation
```

class RdeviationCls

Rdeviation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:FSK:RDEViation
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.fsk.rdeviation.get(result_type = enums.ResultTypeStat.AVG, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the reference deviation of FSK modulated signals.

param result_type

none Measurement deviation for current sweep. AVG Average FSK measurement deviation over several sweeps. RPE Peak FSK measurement deviation over several sweeps. SDEV Standard deviation of FSK measurement deviation. PCTL 95 percentile value of FSK measurement deviation.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.16 Gimbalance

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:GIMBalance
```

class GimbalanceCls

Gimbalance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:GIMBalance
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.gimbalance.get(result_type = enums.ResultTypeStat.AVG, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the Gain Imbalance error measurement of digital demodulation. The output values are the same as those provided in the 'Modulation Accuracy' table .

param result_type

none Gain imbalance error for current sweep AVG Average gain imbalance error over several sweeps RPE Peak gain imbalance error over several sweeps SDEV Standard deviation of gain imbalance error PCTL 95 percentile value of gain imbalance error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.17 IqImbalance**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:IQIMbalance
```

class IqImbalanceCls

IqImbalance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat, window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:IQIMbalance
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.iqImbalance.get(result_type = enums.ResultTypeStat.AVG, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the I/Q imbalance error measurement of digital demodulation.

param result_type

none I/Q imbalance error (for current sweep) AVG Average I/Q imbalance error over several sweeps RPE Peak I/Q imbalance error over several sweeps SDEV Standard deviation of I/Q imbalance error PCTL 95 percentile value of I/Q imbalance error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.18 Merror

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:MERRor
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.merror.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the magnitude error measurement of digital demodulation.

param result_type

none RMS magnitude error of display points of current sweep
 AVG Average of magnitude errors over several sweeps
 PAVG Average of maximum magnitude errors over several sweeps
 PCTL 95% percentile of magnitude error over several sweeps
 PEAK Maximum magnitude errors over all symbols of current sweep
 PPCT 95% percentile of maximum magnitude errors over several sweeps
 PSD Standard deviation of maximum magnitude errors over several sweeps
 RPE Maximum value of magnitude errors over several sweeps
 SDEV Standard deviation of magnitude errors over several sweeps
 TPE Maximum magnitude errors over all display points over several sweeps

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.19 Mpower

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:MPower
```

class MpowerCls

Mpower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:MPower
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.mpower.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```


This command queries the results of the power measurement of digital demodulation.

param result_type

none power measurement (for current sweep) AVG Average of power measurement over several sweeps RPE Peak of power measurement over several sweeps SDEV Standard deviation of power measurement PCTL 95 percentile value of power measurement

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.20 Ooffset

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDemod:STATistic:OOffset
```

class OoffsetCls

Ooffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type*: *ResultTypeStat*, *window*=*Window.Default*, *marker*=*Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDemod:STATistic:OOffset
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.ooffset.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
    ↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the I/Q offset measurement performed for digital demodulation.

param result_type

none Origin offset error (for current sweep) AVG Average origin offset error over several sweeps RPE Peak origin offset error over several sweeps SDEV Standard deviation of origin offset error PCTL 95 percentile value of origin offset error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.21 Perror

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:PERror
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:PERror
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.perror.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the phase error measurement performed for digital demodulation.

param result_type

none 'Phase error' of display points of current sweep AVG Average of phase errors over several sweeps PAVG Average of maximum phase errors over several sweeps PCTL 95% percentile of phase error over several sweeps PEAK Maximum phase error over all symbols of current sweep PPCT 95% percentile of maximum phase errors over several sweeps PSD Standard deviation of maximum phase errors over several sweeps RPE Maximum value of phase error over several sweeps SDEV Standard deviation of phase errors over several sweeps TPE Maximum phase error over all display points over several sweeps

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.22 Qerror

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:QERRor
```

class QerrorCls

Qerror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:QERRor
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
↳ statistic.qerror.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the Quadrature error measurement performed for digital demodulation.

param result_type

none Quadrature error (for current sweep) AVG Average quadrature error over several sweeps RPE Peak quadrature error over several sweeps SDEV Standard deviation of quadrature error PCTL 95 percentile value of quadrature error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.23 Rho

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDemod:STATistic:RHO
```

class RhoCls

Rho commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type*: *ResultTypeStat*, *window*=*Window.Default*, *marker*=*Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDemod:STATistic:RHO
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.rho.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
    ↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the Rho factor measurement performed for digital demodulation.

param result_type

none Rho factor (for current sweep) AVG Average rho factor over several sweeps RPE Peak rho factor over several sweeps SDEV Standard deviation of rho factor PCTL 95 percentile value of rho factor

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.24 Snr

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:SNR
```

class SnrCls

Snr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:SNR
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.snr.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
    ↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the results of the SNR error measurement performed for digital demodulation.

param result_type

none SNR value of display points of current sweep
 AVG Average of SNR values over several sweeps
 PAVG Average of maximum SNR values over several sweeps
 PCTL 95% percentile of SNR value over several sweeps
 PEAK Maximum SNR over all symbols of current sweep
 PPCT 95% percentile of maximum SNR values over several sweeps
 PSD Standard deviation of maximum SNR values over several sweeps
 RPE Maximum value of SNR over several sweeps
 SDEV Standard deviation of SNR values over several sweeps
 TPE Maximum SNR over all display points over several sweeps

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.25 SrError

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMod:STATistic:SRERror
```

class SrErrorCls

SrError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*result_type: ResultTypeStat*, *window=Window.Default*, *marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:DDEMod:STATistic:SRERror
value: float = driver.applications.k70Vsa.calculate.marker.function.ddemod.
    ↳ statistic.srError.get(result_type = enums.ResultTypeStat.AVG, window = repcap.
    ↳ Window.Default, marker = repcap.Marker.Default)
```

This command queries the symbol rate error

param result_type

PEAK | AVG | SDEV | PCTL | TPEak | RPEak | PAVG | PSDev | PPCTL none Symbol rate error (for current sweep) AVG Average symbol rate error over several sweeps RPE Peak symbol rate error over several sweeps SDEV Standard deviation of symbol rate error PCTL 95 percentile value of symbol rate error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

result: No help available

6.1.7.1.12.26 Link

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:LINK
value: bool = driver.applications.k70Vsa.calculate.marker.link.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

With this command markers between several screens can be coupled, i.e. use the same x-value. All screens can be linked with the marker x-value scaled in symbols or time, except those showing the capture buffer. If several capture buffer measurements are visible, their markers are coupled, too.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:LINK
driver.applications.k70Vsa.calculate.marker.link.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

With this command markers between several screens can be coupled, i.e. use the same x-value. All screens can be linked with the marker x-value scaled in symbols or time, except those showing the capture buffer. If several capture buffer measurements are visible, their markers are coupled, too.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.27 Maximum

class MaximumCls

Maximum commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.maximum.clone()
```

Subgroups

6.1.7.1.12.28 Apeak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:APeak
```

class ApeakCls

Apeak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:APeak
driver.applications.k70Vsa.calculate.marker.maximum.apeak.set(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

sets the marker to the largest absolute peak value (maximum or minimum) of the selected trace.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.29 Left**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:LEFT
driver.applications.k70Vsa.calculate.marker.maximum.left.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.30 Next**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:NEXT
driver.applications.k70Vsa.calculate.marker.maximum.next.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.31 Peak**SCPI Commands**

`CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum[:PEAK]
driver.applications.k70Vsa.calculate.marker.maximum.peak.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.32 Right**SCPI Commands**

`CALCulate<Window>:MARKer<Marker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:RIGHT
driver.applications.k70Vsa.calculate.marker.maximum.right.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.33 Mburst**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MBURst:STARt
```

class MburstCls

Mburst commands group definition. 1 total commands, 0 Subgroups, 1 group commands

start(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MBURst:STARt
driver.applications.k70Vsa.calculate.marker.mburst.start(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

This command moves the marker m to the start of the selected result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

start_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.1.7.1.12.34 Minimum**class MinimumCls**

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.minimum.clone()
```

Subgroups

6.1.7.1.12.35 Left

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:LEFT`

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:LEFT
driver.applications.k70Vsa.calculate.marker.minimum.left.set(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.36 Next

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:NEXT
driver.applications.k70Vsa.calculate.marker.minimum.next.set(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.37 Peak**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum[:PEAK]
driver.applications.k70Vsa.calculate.marker.minimum.peak.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.38 Right**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:RIGHT
driver.applications.k70Vsa.calculate.marker.minimum.right.set(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.1.12.39 Search

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:SEARCh`

class SearchCls

Search commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → MarkerRealImagB

```
# SCPI: CALCulate<n>:MARKer<m>:SEARCh
value: enums.MarkerRealImagB = driver.applications.k70Vsa.calculate.marker.
↳search.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command specifies whether the marker search works on the real or the imag trace (for all markers) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

mark_real_imag: REAL | IMAG

set(*mark_real_imag: MarkerRealImagB, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SEARCh
driver.applications.k70Vsa.calculate.marker.search.set(mark_real_imag = enums.
↳MarkerRealImagB.IMAG, window = repcap.Window.Default, marker = repcap.Marker.
↳Default)
```

This command specifies whether the marker search works on the real or the imag trace (for all markers) .

param mark_real_imag

REAL | IMAG

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.40 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
value: bool = driver.applications.k70Vsa.calculate.marker.state.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
driver.applications.k70Vsa.calculate.marker.state.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.41 Trace

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:TRACe`

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
value: float = driver.applications.k70Vsa.calculate.marker.trace.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

trace_number: No help available

set(*trace_number: float, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
driver.applications.k70Vsa.calculate.marker.trace.set(trace_number = 1.0,
↳window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace_number

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.42 X

class XCls

X commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.x.clone()
```

Subgroups

6.1.7.1.12.43 Slimits

class SlimitsCls

Slimits commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.marker.x.slimits.clone()
```

Subgroups

6.1.7.1.12.44 Left

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X:SLIMits:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:LEFT
value: float = driver.applications.k70Vsa.calculate.marker.x.slimits.left.
    .get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command defines the left limit of the marker search range for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

left_limit: No help available

set(left_limit: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:LEFT
driver.applications.k70Vsa.calculate.marker.x.slimits.left.set(left_limit = 1.0,
↪ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command defines the left limit of the marker search range for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param left_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.45 Right

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X:SLIMits:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:RIGHT
value: float = driver.applications.k70Vsa.calculate.marker.x.slimits.right.
↪ get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command defines the right limit of the marker search range for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

right_limit: No help available

set(right_limit: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:RIGHT
driver.applications.k70Vsa.calculate.marker.x.slimits.right.set(right_limit = 1.
↪ 0, window = repcap.Window.Default, marker = repcap.Marker.Default)
```


This command defines the right limit of the marker search range for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param right_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.1.7.1.12.46 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X:SLIMits:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits[:STATe]
value: bool = driver.applications.k70Vsa.calculate.marker.x.slimits.state.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns marker search limits on and off for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits[:STATe]
driver.applications.k70Vsa.calculate.marker.x.slimits.state.set(state = False,
↳window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command turns marker search limits on and off for all markers in all windows. If you perform a measurement in the time domain, this command limits the range of the trace to be analyzed.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.1.7.1.12.47 Y**SCPI Commands**

`CALCulate<Window>:MARKer<Marker>:Y`

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.applications.k70Vsa.calculate.marker.y.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

result: Unit: DBM

6.1.7.1.13 Meas**class MeasCls**

Meas commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.meas.clone()
```

Subgroups

6.1.7.1.13.1 Dirty

SCPI Commands

```
CALCulate<Window>:MEAS:DIRTy
```

class DirtyCls

Dirty commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:MEAS:DIRTy
value: bool = driver.applications.k70Vsa.calculate.meas.dirty.get(window =
↳repcap.Window.Default)
```

Queries the validity of the measurement data, as indicated in the channel bar in manual operation.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | OFF | 0 | 1 OFF | 0 The measurement results are valid. ON | 1 Invalid or inconsistent data is displayed, that is: the trace no longer matches the displayed instrument settings.

6.1.7.1.14 Msra

class MsraCls

Msra commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.msra.clone()
```

Subgroups

6.1.7.1.14.1 Aline

class AlineCls

Aline commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.msra.aline.clone()
```

Subgroups

6.1.7.1.14.2 Show

SCPI Commands

```
CALCulate<Window>:MSRA:ALIne:SHOW
```

class ShowCls

Show commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:MSRA:ALIne:SHOW
value: bool = driver.applications.k70Vsa.calculate.msra.aline.show.get(window = ↵
↵repcap.Window.Default)
```

This command defines whether or not the analysis line is displayed in all time-based windows in all MSRA secondary applications and the MSRA primary application. Note: even if the analysis line display is off, the indication whether or not the currently defined line position lies within the analysis interval of the active secondary application remains in the window title bars.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MSRA:ALIne:SHOW
driver.applications.k70Vsa.calculate.msra.aline.show.set(state = False, window ↵
↵= repcap.Window.Default)
```

This command defines whether or not the analysis line is displayed in all time-based windows in all MSRA secondary applications and the MSRA primary application. Note: even if the analysis line display is off, the indication whether or not the currently defined line position lies within the analysis interval of the active secondary application remains in the window title bars.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.14.3 Value

SCPI Commands

CALCulate<Window>:MSRA:ALINe:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:MSRA:ALINe[:VALue]
value: float = driver.applications.k70Vsa.calculate.msra.aline.value.get(window,
↳repcap.Window.Default)
```

This command defines the position of the analysis line for all time-based windows in all MSRA secondary applications and the MSRA primary application.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

value: No help available

set(*value: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:MSRA:ALINe[:VALue]
driver.applications.k70Vsa.calculate.msra.aline.value.set(value = 1.0, window =
↳repcap.Window.Default)
```

This command defines the position of the analysis line for all time-based windows in all MSRA secondary applications and the MSRA primary application.

param value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.14.4 Window

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.msra.window.clone()
```

Subgroups

6.1.7.1.14.5 Ival

SCPI Commands

```
CALCulate<Window>:MSRA:WINDow:IVAL
```

class IvalCls

Ival commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Int_Start: int: Analysis start = Capture offset time Unit: s
- Int_Stop: int: Analysis end = capture offset + capture time Unit: s

get(window=Window.Default) → GetStruct

```
# SCPI: CALCulate<n>:MSRA:WINDow:IVAL
value: GetStruct = driver.applications.k70Vsa.calculate.msra.window.ival.
↪get(window = repcap.Window.Default)
```

Returns the current analysis interval for applications in MSRA operating mode.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

structure: for return value, see the help for GetStruct structure arguments.

6.1.7.1.15 Statistics

SCPI Commands

```
CALCulate<Window>:STATistics:PRESet
```

class StatisticsCls

Statistics commands group definition. 7 total commands, 3 Subgroups, 1 group commands

preset(window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:PRESet
driver.applications.k70Vsa.calculate.statistics.preset(window = repcap.Window.
↪Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

preset_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.statistics.clone()
```

Subgroups

6.1.7.1.15.1 CumulativeDistribFnc

class CumulativeDistribFncCls

CumulativeDistribFnc commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.statistics.cumulativeDistribFnc.clone()
```

Subgroups

6.1.7.1.15.2 State

SCPI Commands

```
CALCulate<Window>:STATistics:CCDF:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:STATistics:CCDF[:STATe]
value: bool = driver.applications.k70Vsa.calculate.statistics.
    cumulativeDistribFnc.state.get(window = repcap.Window.Default)
```

This command switches the measurement of the statistical distribution of magnitude, phase or frequency values on or off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:CCDF[:STATe]
driver.applications.k70Vsa.calculate.statistics.cumulativeDistribFnc.state.
↪set(state = False, window = repcap.Window.Default)
```

This command switches the measurement of the statistical distribution of magnitude, phase or frequency values on or off.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.15.3 Mode**SCPI Commands**

```
CALCulate<Window>:STATistics:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → StatisticMode

```
# SCPI: CALCulate<n>:STATistics:MODE
value: enums.StatisticMode = driver.applications.k70Vsa.calculate.statistics.
↪mode.get(window = repcap.Window.Default)
```

This command defines whether only the symbol points or all points are considered for the statistical calculations.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

statistic_mode: SONLy | INFinite SONLy Symbol points only are used INFinite All points are used

set(statistic_mode: StatisticMode, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:MODE
driver.applications.k70Vsa.calculate.statistics.mode.set(statistic_mode = enums.
↪StatisticMode.INFinite, window = repcap.Window.Default)
```

This command defines whether only the symbol points or all points are considered for the statistical calculations.

param statistic_mode

SONLy | INFinite SONLy Symbol points only are used INFinite All points are used

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.15.4 Scale**class ScaleCls**

Scale commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.statistics.scale.clone()
```

Subgroups**6.1.7.1.15.5 X****class XCls**

X commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.statistics.scale.x.clone()
```

Subgroups**6.1.7.1.15.6 Bcount****SCPI Commands**

```
CALCulate<Window>:STATistics:SCALe:X:BCOunt
```

class BcountCls

Bcount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALe:X:BCOunt
value: float = driver.applications.k70Vsa.calculate.statistics.scale.x.bcount.
    ↪get(window = repcap.Window.Default)
```

This command defines the number of columns for the statistical distribution.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
stat_nof_columns: Range: 2 to 1024, Unit: none
set(stat_nof_columns: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:X:BCount
driver.applications.k70Vsa.calculate.statistics.scale.x.bcount.set(stat_nof_
↪columns = 1.0, window = repcap.Window.Default)
```

This command defines the number of columns for the statistical distribution.

param stat_nof_columns
Range: 2 to 1024, Unit: none
param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.15.7 Y

class YCls

Y commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.statistics.scale.y.clone()
```

Subgroups

6.1.7.1.15.8 Lower

SCPI Commands

```
CALCulate<Window>:STATistics:SCALE:Y:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:LOWer
value: float = driver.applications.k70Vsa.calculate.statistics.scale.y.lower.
↪get(window = repcap.Window.Default)
```

This command defines the lower vertical limit of the diagram.

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)
return
start: No help available

set(*start*: float, *window*=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:LOWer
driver.applications.k70Vsa.calculate.statistics.scale.y.lower.set(start = 1.0,
↪window = repcap.Window.Default)
```

This command defines the lower vertical limit of the diagram.

param start

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.15.9 Unit

SCPI Commands

CALCulate<Window>:STATistics:SCALE:Y:UNIT

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default) → ScaleYaxisUnit

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UNIT
value: enums.ScaleYaxisUnit = driver.applications.k70Vsa.calculate.statistics.
↪scale.y.unit.get(window = repcap.Window.Default)
```

This command selects the unit of the y-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: PCT | ABS

set(*unit*: ScaleYaxisUnit, *window*=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UNIT
driver.applications.k70Vsa.calculate.statistics.scale.y.unit.set(unit = enums.
↪ScaleYaxisUnit.ABS, window = repcap.Window.Default)
```

This command selects the unit of the y-axis.

param unit

PCT | ABS

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.15.10 Upper

SCPI Commands

`CALCulate<Window>:STATistics:SCALE:Y:UPPer`

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UPPer
value: float = driver.applications.k70Vsa.calculate.statistics.scale.y.upper.
↪get(window = repcap.Window.Default)
```

This command defines the upper vertical limit of the diagram.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

stop: No help available

set(stop: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UPPer
driver.applications.k70Vsa.calculate.statistics.scale.y.upper.set(stop = 1.0, ↪
↪window = repcap.Window.Default)
```

This command defines the upper vertical limit of the diagram.

param stop

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.16 TIAbs

class TIAbsCls

TIAbs commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.tIAbs.clone()
```

Subgroups

6.1.7.1.16.1 State

SCPI Commands

CALCulate<Window>:TLABs:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:TLABs:STATe
value: bool = driver.applications.k70Vsa.calculate.tlAbs.state.get(window = ↵
↵repcap.Window.Default)
```

Displays an absolute vertical line in the specified window. This command is only available for eye diagrams.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:TLABs:STATe
driver.applications.k70Vsa.calculate.tlAbs.state.set(state = False, window = ↵
↵repcap.Window.Default)
```

Displays an absolute vertical line in the specified window. This command is only available for eye diagrams.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.16.2 Value

SCPI Commands

CALCulate<Window>:TLABs:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:TLABs[:VALue]
value: float = driver.applications.k70Vsa.calculate.tlAbs.value.get(window = ↵
↵repcap.Window.Default)
```

Defines or queries the x-value of the absolute vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line is already available in the diagram (see method RsFswp.Applications. K70_Vsa.Calculate.TlAbs.State.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

vert_line_abs_pos: X-value of the absolute vertical line. Unit: SYMB

set(vert_line_abs_pos: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:TLABs[:VALue]
driver.applications.k70Vsa.calculate.tlAbs.value.set(vert_line_abs_pos = 1.0,
↪window = repcap.Window.Default)
```

Defines or queries the x-value of the absolute vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line is already available in the diagram (see method RsFswp.Applications. K70_Vsa.Calculate.TlAbs.State.set) .

param vert_line_abs_pos

X-value of the absolute vertical line. Unit: SYMB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.17 TlRel

class TlRelCls

TlRel commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.tlRel.clone()
```

Subgroups

6.1.7.1.17.1 State

SCPI Commands

```
CALCulate<Window>:TLRel:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:TLRel:STATE
value: bool = driver.applications.k70Vsa.calculate.tlRel.state.get(window = ↵
↵repcap.Window.Default)
```

Displays a relative vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line is already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.TlAbs.State.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:TLRel:STATE
driver.applications.k70Vsa.calculate.tlRel.state.set(state = False, window = ↵
↵repcap.Window.Default)
```

Displays a relative vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line is already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.TlAbs.State.set) .

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.17.2 Value

SCPI Commands

```
CALCulate<Window>:TLRel:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:TLRel[:VALue]
value: float = driver.applications.k70Vsa.calculate.tlRel.value.get(window = ↵
↵repcap.Window.Default)
```

Defines or queries the x-value of the relative vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line and a relative vertical line are already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.TlAbs.State.set and method RsFswp.Applications.K70_Vsa.Calculate.TlRel.State.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

line_rel_pos_rel: Relative distance of the second vertical line to the first (absolute) vertical line. Unit: SYMB

set(line_rel_pos_rel: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:TLRel[:VALue]
driver.applications.k70Vsa.calculate.tlRel.value.set(line_rel_pos_rel = 1.0,
↪window = repcap.Window.Default)
```

Defines or queries the x-value of the relative vertical line in the specified window. This command is only available for eye diagrams, and only if an absolute vertical line and a relative vertical line are already available in the same diagram (see method RsFswp.Applications.K70_Vsa.Calculate.TlAbs.State.set and method RsFswp.Applications.K70_Vsa.Calculate.TlRel.State.set) .

param line_rel_pos_rel

Relative distance of the second vertical line to the first (absolute) vertical line. Unit: SYMB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.18 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.applications.k70Vsa.calculate.trace.repcap_trace_get()
driver.applications.k70Vsa.calculate.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 5 total commands, 3 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.trace.clone()
```

Subgroups

6.1.7.1.18.1 Adjust

class AdjustCls

Adjust commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.trace.adjust.clone()
```

Subgroups

6.1.7.1.18.2 Alignment

class AlignmentCls

Alignment commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.trace.adjust.alignment.clone()
```

Subgroups

6.1.7.1.18.3 Default

SCPI Commands

```
CALCulate<Window>:TRACe<Trace>:ADJust:ALIGnment:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → AdjustAlignment

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust:ALIGnment[:DEFault]
value: enums.AdjustAlignment = driver.applications.k70Vsa.calculate.trace.
↪adjust.alignment.default.get(window = repcap.Window.Default, trace = repcap.
↪Trace.Default)
```

This command defines where the reference point is to appear in the result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

alignment: LEFT | CENTER | RIGHT
LEFT The reference point is at the start of the result range. CENTER The reference point is in the middle of the result range. RIGHT The reference point is displayed at the end of the result range.

set(alignment: AdjustAlignment, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust:ALIGnment[:DEFault]
driver.applications.k70Vsa.calculate.trace.adjust.alignment.default.
↪set(alignment = enums.AdjustAlignment.CENTer, window = repcap.Window.Default, ↪
↪trace = repcap.Trace.Default)
```

This command defines where the reference point is to appear in the result range.

param alignment

LEFT | CENTer | RIGHt LEFT The reference point is at the start of the result range. CENTer The reference point is in the middle of the result range. RIGHt The reference point is displayed at the end of the result range.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.1.7.1.18.4 Offset

SCPI Commands

CALCulate<Window>:TRACe<Trace>:ADJust:ALIGnment:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust:ALIGnment:OFFSet
value: float = driver.applications.k70Vsa.calculate.trace.adjust.alignment.
↪offset.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

fit_offset: Unit: SYM

set(fit_offset: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust:ALIGnment:OFFSet
driver.applications.k70Vsa.calculate.trace.adjust.alignment.offset.set(fit_
↪offset = 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param fit_offset

Unit: SYM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.1.18.5 Value

SCPI Commands

```
CALCulate<Window>:TRACe<Trace>:ADJust:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → TraceReference

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust[:VALue]
value: enums.TraceReference = driver.applications.k70Vsa.calculate.trace.adjust.
↪ value.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the reference point for the display.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

reference: TRIGger | BURSt | PATtern TRIGger The reference point is defined by the start of the capture buffer. BURSt The reference point is defined by the start/center/end of the burst. PATtern The instrument selects the reference point and the alignment.

set(*reference: TraceReference, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: CALCulate<n>:TRACe<t>:ADJust[:VALue]
driver.applications.k70Vsa.calculate.trace.adjust.value.set(reference = enums.
↪ TraceReference.BURSt, window = repcap.Window.Default, trace = repcap.Trace.
↪ Default)
```

This command defines the reference point for the display.

param reference

TRIGger | BURSt | PATtern TRIGger The reference point is defined by the start of the capture buffer. BURSt The reference point is defined by the start/center/end of the burst. PATtern The instrument selects the reference point and the alignment.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.1.7.1.18.6 Symbols**SCPI Commands**

CALCulate<Window>:TRACe<Trace>:SYMBols
--

class SymbolsCls

Symbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → SymbolSelection

<pre># SCPI: CALCulate<n>:TRACe<t>:SYMBols value: enums.SymbolSelection = driver.applications.k70Vsa.calculate.trace. ↳symbols.get(window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>
--

This commands selects which symbols are displayed by a trace (in a constellation graph with 2 modulations). For method RsFswp.Display.Window.Subwindow.Trace.Mode.set View, the symbol selection cannot be changed. It remains set to the value that was most recently set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

symbol_selection: ALL | PATTeRn | DATA ALL Trace consists of constellation points for all symbols PATTeRn Trace consists of only pattern symbols DATA Trace consists of only data symbols

set(*symbol_selection: SymbolSelection, window=Window.Default, trace=Trace.Default*) → None

<pre># SCPI: CALCulate<n>:TRACe<t>:SYMBols driver.applications.k70Vsa.calculate.trace.symbols.set(symbol_selection = enums. ↳SymbolSelection.ALL, window = repcap.Window.Default, trace = repcap.Trace. ↳Default)</pre>

This commands selects which symbols are displayed by a trace (in a constellation graph with 2 modulations). For method RsFswp.Display.Window.Subwindow.Trace.Mode.set View, the symbol selection cannot be changed. It remains set to the value that was most recently set.

param symbol_selection

ALL | PATTeRn | DATA ALL Trace consists of constellation points for all symbols PATTeRn Trace consists of only pattern symbols DATA Trace consists of only data symbols

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.1.18.7 Value**SCPI Commands**

CALCulate<Window>:TRACe<Trace>:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TraceRefType

<pre># SCPI: CALCulate<n>:TRACe<t>[:VALue] value: enums.TraceRefType = driver.applications.k70Vsa.calculate.trace.value. ↪get(window = repcap.Window.Default, trace = repcap.Trace.Default)</pre>

This commands selects the signal to be used as the data source for a trace. For method RsFswp.Display.Window.Subwindow.Trace.Mode.setView, the data source to be evaluated cannot be changed. It remains set to the value that was most recently set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

trace_ref_type: MEAS | REF | ERRor | TCAP MEAS Measurement signal REF Reference signal ERR Error TCAP Capture buffer

set(trace_ref_type: TraceRefType, window=Window.Default, trace=Trace.Default) → None

<pre># SCPI: CALCulate<n>:TRACe<t>[:VALue] driver.applications.k70Vsa.calculate.trace.value.set(trace_ref_type = enums. ↪TraceRefType.ERROR, window = repcap.Window.Default, trace = repcap.Trace. ↪Default)</pre>
--

This commands selects the signal to be used as the data source for a trace. For method RsFswp.Display.Window.Subwindow.Trace.Mode.setView, the data source to be evaluated cannot be changed. It remains set to the value that was most recently set.

param trace_ref_type

MEAS | REF | ERRor | TCAP MEAS Measurement signal REF Reference signal ERR Error TCAP Capture buffer

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.1.19 Unit**class UnitCls**

Unit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.unit.clone()
```

Subgroups**6.1.7.1.19.1 Angle****SCPI Commands**

```
CALCulate<Window>:UNIT:ANGLE
```

class AngleCls

Angle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AngleUnit

```
# SCPI: CALCulate<n>:UNIT:ANGLE
value: enums.AngleUnit = driver.applications.k70Vsa.calculate.unit.angle.
↪get(window = repcap.Window.Default)
```

This command selects the global unit for phase results.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

unit: DEG | RAD

set(unit: AngleUnit, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNIT:ANGLE
driver.applications.k70Vsa.calculate.unit.angle.set(unit = enums.AngleUnit.DEG, ↪
↪window = repcap.Window.Default)
```

This command selects the global unit for phase results.

param unit

DEG | RAD

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.19.2 Power**SCPI Commands**

CALCulate<Window>:UNIT:POWer

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → PowerUnitDdem

<pre># SCPI: CALCulate<n>:UNIT:POWer value: enums.PowerUnitDdem = driver.applications.k70Vsa.calculate.unit.power. ↳get(window = repcap.Window.Default)</pre>

This command selects the unit of the y-axis. The unit applies to all power-based measurement windows with absolute values.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: DBM | V | A | W | DBPW | WATT | DBUV | DBMV | VOLT | DBUA | AMPere

set(unit: PowerUnitDdem, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:UNIT:POWer driver.applications.k70Vsa.calculate.unit.power.set(unit = enums.PowerUnitDdem. ↳DBM, window = repcap.Window.Default)</pre>
--

This command selects the unit of the y-axis. The unit applies to all power-based measurement windows with absolute values.

param unit

DBM | V | A | W | DBPW | WATT | DBUV | DBMV | VOLT | DBUA | AMPere

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.1.20 X**class XCls**

X commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.x.clone()
```

Subgroups

6.1.7.1.20.1 Unit

class UnitCls

Unit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.x.unit.clone()
```

Subgroups

6.1.7.1.20.2 Time

SCPI Commands

```
CALCulate<Window>:X:UNIT:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*) → TimeLimitUnit

```
# SCPI: CALCulate<n>:X:UNIT:TIME
value: enums.TimeLimitUnit = driver.applications.k70Vsa.calculate.x.unit.time.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: S | SYM

set(unit: TimeLimitUnit, window=*Window.Default*) → None

```
# SCPI: CALCulate<n>:X:UNIT:TIME
driver.applications.k70Vsa.calculate.x.unit.time.set(unit = enums.TimeLimitUnit.
↳S, window = repcap.Window.Default)
```

No command help available

param unit

S | SYM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.1.7.1.21 Y

class YCls

Y commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.y.clone()
```

Subgroups

6.1.7.1.21.1 Unit

class UnitCls

Unit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.calculate.y.unit.clone()
```

Subgroups

6.1.7.1.21.2 Time

SCPI Commands

```
CALCulate<Window>:Y:UNIT:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → TimeLimitUnit

```
# SCPI: CALCulate<n>:Y:UNIT:TIME
value: enums.TimeLimitUnit = driver.applications.k70Vsa.calculate.y.unit.time.
←get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: S | SYM

set(unit: TimeLimitUnit, window=Window.Default) → None

```
# SCPI: CALCulate<n>:Y:UNIT:TIME
driver.applications.k70Vsa.calculate.y.unit.time.set(unit = enums.TimeLimitUnit.
↪S, window = repcap.Window.Default)
```

No command help available

param unit

S | SYM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.1.7.2 Display

class DisplayCls

Display commands group definition. 22 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.clone()
```

Subgroups

6.1.7.2.1 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.display.window.repcap_window_get()
driver.applications.k70Vsa.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 22 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.clone()
```

Subgroups

6.1.7.2.1.1 Item

class ItemCls

Item commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.item.clone()
```

Subgroups

6.1.7.2.1.2 Line

class LineCls

Line commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.item.line.clone()
```

Subgroups

6.1.7.2.1.3 Value

SCPI Commands

```
DISPlay:WINDow<Window>:ITEM:LINE:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → SingleValue

```
# SCPI: DISPlay[:WINDow<n>]:ITEM[:LINE][:VALue]
value: enums.SingleValue = driver.applications.k70Vsa.display.window.item.line.
    ↪ value.get(window = repcap.Window.Default)
```

This commands switches between the whole ‘Result Summary’ and the diagram showing only a single value, e.g. the EVM RMS value as a bargraph. The same parameters are available as those for which modulation accuracy limits can be defined (see ‘Limit Value’).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

return

single_value: ALL | EVMR | EVMP | PERM | PEP | MERM | MEP | CFER | RHO | IQOF | FERM | FEP | FDER ALL Complete ‘Result Summary’ EVMR RMS EVM EVMP Peak EVM PERM RMS Phase error PEP Peak phase error MERM RMS Magnitude error MEP Peak magnitude error CFER Carrier frequency error RHO RHO IQOF I/Q offset FERM RMS frequency error FEP Peak frequency error FDER FSK deviation error

set(single_value: SingleValue, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:ITEM[:LINE][:VALUE]
driver.applications.k70Vsa.display.window.item.line.value.set(single_value =
enums.SingleValue.ALL, window = repcap.Window.Default)
```

This commands switches between the whole ‘Result Summary’ and the diagram showing only a single value, e.g. the EVM RMS value as a bargraph. The same parameters are available as those for which modulation accuracy limits can be defined (see ‘Limit Value’).

param single_value

ALL | EVMR | EVMP | PERM | PEP | MERM | MEP | CFER | RHO | IQOF | FERM | FEP | FDER ALL Complete ‘Result Summary’ EVMR RMS EVM EVMP Peak EVM PERM RMS Phase error PEP Peak phase error MERM RMS Magnitude error MEP Peak magnitude error CFER Carrier frequency error RHO RHO IQOF I/Q offset FERM RMS frequency error FEP Peak frequency error FDER FSK deviation error

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

6.1.7.2.1.4 Prate

class PrateCls

Prate commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.prate.clone()
```

Subgroups

6.1.7.2.1.5 Auto

SCPI Commands

```
DISPlay:WINDow<Window>:PRATe:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AutoManualMode

```
# SCPI: DISPlay[:WINDow<n>]:PRATe:AUTO
value: enums.AutoManualMode = driver.applications.k70Vsa.display.window.prate.
↳ auto.get(window = repcap.Window.Default)
```

Defines the number of display points that are displayed per symbol automatically, i.e. according to [SENSe:]DDEMod:PRATe. To define a different number of points per symbol for display, use the MANUAL parameter and the method RsFswp. Applications.K70_Vsa.Display.Window.Prate.Value.set command.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

display_pps_mode: AUTO | MANUAL

set(display_pps_mode: AutoManualMode, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:PRATe:AUTO
driver.applications.k70Vsa.display.window.prate.auto.set(display_pps_mode =
↳ enums.AutoManualMode.AUTO, window = repcap.Window.Default)
```

Defines the number of display points that are displayed per symbol automatically, i.e. according to [SENSe:]DDEMod:PRATe. To define a different number of points per symbol for display, use the MANUAL parameter and the method RsFswp. Applications.K70_Vsa.Display.Window.Prate.Value.set command.

param display_pps_mode

AUTO | MANUAL

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.7.2.1.6 Value

SCPI Commands

```
DISPlay:WINDow<Window>:PRATe:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:PRATe[:VALue]
value: float = driver.applications.k70Vsa.display.window.prate.value.get(window=
↳ repcap.Window.Default)
```

This command determines the number of points to be displayed per symbol if manual mode is selected (see method RsFswp. Applications.K70_Vsa.Display.Window.Prate.Auto.set) . This command is not available for result displays based on the capture buffer; in this case, the displayed points per symbol are defined by the sample rate ([SENSe:]DDEMod:PRATe command) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

display_pps: 1, 2, 4, 8, 16 or 32 1 only the symbol time instants are displayed 2, 4, 8, 16, 32 more points are displayed than symbols

set(display_pps: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:PRATe[:VALue]
driver.applications.k70Vsa.display.window.prate.value.set(display_pps = 1.0,
↳ window = repcap.Window.Default)
```

This command determines the number of points to be displayed per symbol if manual mode is selected (see method RsFswp. Applications.K70_Vsa.Display.Window.Prate.Auto.set) . This command is not available for result displays based on the capture buffer; in this case, the displayed points per symbol are defined by the sample rate ([SENSe:]DDEMod:PRATe command) .

param display_pps

1, 2, 4, 8, 16 or 32 1 only the symbol time instants are displayed 2, 4, 8, 16, 32 more points are displayed than symbols

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.7.2.1.7 Size

SCPI Commands

```
DISPlay:WINDow<Window>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → Size

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
value: enums.Size = driver.applications.k70Vsa.display.window.size.get(window =
↳ repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method

RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

size: LARGE Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

set(size: Size, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
driver.applications.k70Vsa.display.window.size.set(size = enums.Size.LARGE,
↪window = repcap.Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set command (see method RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set) .

param size

LARGE Maximizes the selected window to full screen. Other windows are still active in the background. SMALL Reduces the size of the selected window to its original size. If more than one measurement window was displayed originally, these are visible again.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.1.7.2.1.8 State

SCPI Commands

DISPlay:WINDow<Window>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:STATe
value: bool = driver.applications.k70Vsa.display.window.state.get(window =
↪repcap.Window.Default)
```

Activates or deactivates the specified window. Note that this command is maintained for compatibility reasons only. Use the LAYout commands for new remote control programs (see 'Working with windows in the display') .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:STATe
driver.applications.k70Vsa.display.window.state.set(state = False, window =
↳repcap.Window.Default)
```

Activates or deactivates the specified window. Note that this command is maintained for compatibility reasons only. Use the LAYout commands for new remote control programs (see ‘Working with windows in the display’).

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

6.1.7.2.1.9 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.applications.k70Vsa.display.window.trace.repcap_trace_get()
driver.applications.k70Vsa.display.window.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 17 total commands, 6 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.clone()
```

Subgroups

6.1.7.2.1.10 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TraceModeF


```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
value: enums.TraceModeF = driver.applications.k70Vsa.display.window.trace.mode.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: No help available

set(mode: TraceModeF, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
driver.applications.k70Vsa.display.window.trace.mode.set(mode = enums.
↳TraceModeF.AVERage, window = repcap.Window.Default, trace = repcap.Trace.
↳Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.11 Preset

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:PRESet
```

class PresetCls

Preset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TracePresetType

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
value: enums.TracePresetType = driver.applications.k70Vsa.display.window.trace.
↳preset.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

result_type: No help available

set(result_type: TracePresetType, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:PRESet
driver.applications.k70Vsa.display.window.trace.preset.set(result_type = enums.
↳ TracePresetType.ALL, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

No command help available

param result_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.12 State

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
value: bool = driver.applications.k70Vsa.display.window.trace.state.get(window,
↳ repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
driver.applications.k70Vsa.display.window.trace.state.set(state = False, window↵
↵= repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.13 Symbol

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SYMBol
```

class SymbolCls

Symbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → TraceSymbols

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SYMBol
value: enums.TraceSymbols = driver.applications.k70Vsa.display.window.trace.
↵symbol.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command enables the display of the decision instants (time when the signals occurred) as dots on the trace.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

symbols: ON | OFF | 0 | 1 OFF | 0 Symbols are displayed. ON | 1 Symbols are not displayed.

set(symbols: TraceSymbols, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SYMBol
driver.applications.k70Vsa.display.window.trace.symbol.set(symbols = enums.
↵TraceSymbols.BARS, window = repcap.Window.Default, trace = repcap.Trace.
↵Default)
```

This command enables the display of the decision instants (time when the signals occurred) as dots on the trace.

param symbols

ON | OFF | 0 | 1 OFF | 0 Symbols are displayed. ON | 1 Symbols are not displayed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.14 X**class XCls**

X commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.x.clone()
```

Subgroups**6.1.7.2.1.15 Scale****class ScaleCls**

Scale commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.x.scale.clone()
```

Subgroups**6.1.7.2.1.16 Pdivision****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALE:PDIVision
```

class PdivisionCls

Pdivision commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALE]:PDIVision
value: float = driver.applications.k70Vsa.display.window.trace.x.scale.
↪ pdivision.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the scaling of the x-axis for statistical result displays. For all other result displays, this command is only available as a query.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

pdiv: Defines the range per division (total range = 10*PDiv)

set(pdiv: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALE]:PDIVision
driver.applications.k70Vsa.display.window.trace.x.scale.pdivision.set(pdiv = 1.
↪ 0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the scaling of the x-axis for statistical result displays. For all other result displays, this command is only available as a query.

param pdiv

Defines the range per division (total range = 10*PDiv)

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.17 RefPosition

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALE:RPOSition
```

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALE]:RPOSition
value: float = driver.applications.k70Vsa.display.window.trace.x.scale.
↪ refPosition.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the position of the reference value for the X axis. Setting the position of the reference value is possible only for statistical result displays. All other result displays support the query only.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

rpos: numeric_value Unit: PCT

set(rpos: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]:RPOSition
driver.applications.k70Vsa.display.window.trace.x.scale.refPosition.set(rpos = 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the position of the reference value for the X axis. Setting the position of the reference value is possible only for statistical result displays. All other result displays support the query only.

param rpos

numeric_value Unit: PCT

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.18 Rvalue

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALe:RVALue
```

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]:RVALue
value: float = driver.applications.k70Vsa.display.window.trace.x.scale.rvalue.
get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the reference value for the x-axis for statistical result displays. For all other result displays, this command is only available as a query.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

rval: Reference value for the x-axis

set(rval: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]:RVALue
driver.applications.k70Vsa.display.window.trace.x.scale.rvalue.set(rval = 1.0,
↪window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the reference value for the x-axis for statistical result displays. For all other result displays, this command is only available as a query.

param rval

Reference value for the x-axis

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.19 Start**SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALe:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → int

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]:START
value: int = driver.applications.k70Vsa.display.window.trace.x.scale.start.
↪get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command queries the first value of the x-axis in the specified window in symbols or time, depending on the unit setting for the x-axis. Note: using the method RsFswp.Applications.K70_Vsa.Calculate.Trace.Adjust.Alignment.Offset.set command, the burst is shifted in the diagram; the x-axis thus no longer begins on the left at 0 symbols but at a selectable value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

start: No help available

6.1.7.2.1.20 Stop

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALE:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → int

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALE]:STOP
value: int = driver.applications.k70Vsa.display.window.trace.x.scale.stop.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command queries the last value of the x-axis in the specified window in symbols or time, depending on the unit setting for the x-axis. Note: If the burst is shifted (using the CALC:TRAC:ALIG commands) the x-axis no longer begins at 0 symbols on the left, but at a user-defined value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

stop: No help available

6.1.7.2.1.21 Voffset

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:X:SCALE:VOFFset
```

class VoffsetCls

Voffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALE]:VOFFset
value: float = driver.applications.k70Vsa.display.window.trace.x.scale.voffset.
↳get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines an offset to numbering of the symbols (Except capture buffer) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

voffset: Range: -100000 to 100000, Unit: none

set(voffset: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:X[:SCALe]:VOFFset
driver.applications.k70Vsa.display.window.trace.x.scale.voffset.set(voffset = 1.
↪0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines an offset to numbering of the symbols (Except capture buffer) .

param voffset

Range: -100000 to 100000, Unit: none

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.22 Y

class YCls

Y commands group definition. 7 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.y.clone()
```

Subgroups

6.1.7.2.1.23 Scale

class ScaleCls

Scale commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.y.scale.clone()
```

Subgroups

6.1.7.2.1.24 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.y.scale.auto.clone()
```

Subgroups

6.1.7.2.1.25 All

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:AUTO:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:AUTO:ALL
driver.applications.k70Vsa.display.window.trace.y.scale.auto.all.set(window = re
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

Automatic scaling of the y-axis is performed once in all windows, then switched off again.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

set_with_opc(window=Window.Default, trace=Trace.Default, opc_timeout_ms: int = -1) → None

6.1.7.2.1.26 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → ReferenceMode

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MODE
value: enums.ReferenceMode = driver.applications.k70Vsa.display.window.trace.y.
↳ scale.mode.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: No help available

set(*mode: ReferenceMode, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MODE
driver.applications.k70Vsa.display.window.trace.y.scale.mode.set(mode = enums.
↳ ReferenceMode.ABSolute, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.27 Pdivision

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:PDIVision
```

class PdivisionCls

Pdivision commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE][:PDIVision]
value: float = driver.applications.k70Vsa.display.window.trace.y.scale.
↳ pdivision.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

range_py: No help available

set(range_py: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe][:PDIVision]
driver.applications.k70Vsa.display.window.trace.y.scale.pdivision.set(range_py=
↪ 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param range_py

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.28 RefLevel

class RefLevelCls

RefLevel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.display.window.trace.y.scale.refLevel.clone()
```

Subgroups

6.1.7.2.1.29 Offset

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RLEVel:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet
value: float = driver.applications.k70Vsa.display.window.trace.y.scale.refLevel.
↳ offset.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

level_offset: No help available

set(level_offset: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet
driver.applications.k70Vsa.display.window.trace.y.scale.refLevel.offset.
↳ set(level_offset = 1.0, window = repcap.Window.Default, trace = repcap.Trace.
↳ Default)
```

No command help available

param level_offset

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.30 RefPosition

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RPOSition
```

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RPOSition
value: float = driver.applications.k70Vsa.display.window.trace.y.scale.
↳ refPosition.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

ref_position: No help available

set(ref_position: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RPOSition
driver.applications.k70Vsa.display.window.trace.y.scale.refPosition.set(ref_
↪ position = 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param ref_position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.31 Rvalue

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RVALue
```

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RVALue
value: float = driver.applications.k70Vsa.display.window.trace.y.scale.rvalue.
↪ get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

ref_value: No help available

set(*ref_value*: float, *window*=Window.Default, *trace*=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RVALue
driver.applications.k70Vsa.display.window.trace.y.scale.rvalue.set(ref_value = 1.0, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param ref_value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.2.1.32 Spacing

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SPACing
```

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default, *trace*=Trace.Default) → ScalingMode

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y:SPACing
value: enums.ScalingMode = driver.applications.k70Vsa.display.window.trace.y.spacing.get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

unit: No help available

set(*unit*: ScalingMode, *window*=Window.Default, *trace*=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y:SPACing
driver.applications.k70Vsa.display.window.trace.y.spacing.set(unit = enums.ScalingMode.LINear, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param unit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.1.7.3 FormatPy

class FormatPyCls

FormatPy commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.formatPy.clone()
```

Subgroups

6.1.7.3.1 Dexport

class DexportCls

Dexport commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.formatPy.dexport.clone()
```

Subgroups

6.1.7.3.1.1 Dseparator

SCPI Commands

```
FORMat:DEXPort:DSEPARATOR
```

class DseparatorCls

Dseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Separator

```
# SCPI: FORMat:DEXPort:DSEPARATOR
value: enums.Separator = driver.applications.k70Vsa.formatPy.dexport.dseparator.
↳ get()
```


This command selects the decimal separator for data exported in ASCII format.

return

separator: POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05.

POINT Uses a point as decimal separator, e.g. 4.05.

set(separator: Separator) → None

```
# SCPI: FORMat:DEXPort:DSEParator
driver.applications.k70Vsa.formatPy.dexport.dseparator.set(separator = enums.
↪Separator.COMMa)
```

This command selects the decimal separator for data exported in ASCII format.

param separator

POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05. POINT Uses

a point as decimal separator, e.g. 4.05.

6.1.7.3.1.2 Header

SCPI Commands

FORMat:DEXPort:HEADer

class HeaderCls

Header commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: FORMat:DEXPort:HEADer
value: bool = driver.applications.k70Vsa.formatPy.dexport.header.get()
```

This command defines if a file header (including start frequency, sweep time, detector, etc.) is created or not. A short header with the instrument model, the version and the date is always transferred.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: FORMat:DEXPort:HEADer
driver.applications.k70Vsa.formatPy.dexport.header.set(state = False)
```

This command defines if a file header (including start frequency, sweep time, detector, etc.) is created or not. A short header with the instrument model, the version and the date is always transferred.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.3.1.3 Mode

SCPI Commands

FORMat:DEXPort:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DataExportMode

```
# SCPI: FORMat:DEXPort:MODE
value: enums.DataExportMode = driver.applications.k70Vsa.formatPy.dexport.mode.
↪get()
```

This command defines which data are transferred, raw I/Q data or trace data.

return
mode: RAW | TRACe

set(mode: DataExportMode) → None

```
# SCPI: FORMat:DEXPort:MODE
driver.applications.k70Vsa.formatPy.dexport.mode.set(mode = enums.
↪DataExportMode.RAW)
```

This command defines which data are transferred, raw I/Q data or trace data.

param mode
RAW | TRACe

6.1.7.4 Initiate

class InitiateCls

Initiate commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.initiate.clone()
```

Subgroups

6.1.7.4.1 ConMeas

SCPI Commands

INITiate:CONMeas

class ConMeasCls

ConMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:CONMeas
driver.applications.k70Vsa.initiate.conMeas.set()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:CONMeas
driver.applications.k70Vsa.initiate.conMeas.set_with_opc()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.4.2 Continuous

SCPI Commands

```
INITiate:CONTinuous
```

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INITiate:CONTinuous
value: bool = driver.applications.k70Vsa.initiate.continuous.get()
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with *OPC, *OPC? or *WAI. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

return

state: ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

set(state: bool) → None

```
# SCPI: INITiate:CONTinuous
driver.applications.k70Vsa.initiate.continuous.set(state = False)
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with ***OPC**, ***OPC?** or ***WAI**. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

param state

ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

6.1.7.4.3 Immediate

SCPI Commands

```
INITiate:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k70Vsa.initiate.immediate.set()
```

This command starts a (single) new measurement. For a statistics count > 0, this means a restart of the corresponding number of measurements. With trace mode MAXHold, MINHold and AVERage, the previous results are reset on restarting the measurement. You can synchronize to the end of the measurement with ***OPC**, ***OPC?** or ***WAI**. For details on synchronization see Remote control via SCPI.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate[:IMMediate]
driver.applications.k70Vsa.initiate.immediate.set_with_opc()
```

This command starts a (single) new measurement. For a statistics count > 0, this means a restart of the corresponding number of measurements. With trace mode MAXHold, MINHold and AVERage, the previous results are reset on restarting the measurement. You can synchronize to the end of the measurement with ***OPC**, ***OPC?** or ***WAI**. For details on synchronization see Remote control via SCPI.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.4.4 RefMeas

SCPI Commands

```
INITiate:REFMeas
```

class RefMeasCls

RefMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:REFMeas
driver.applications.k70Vsa.initiate.refMeas.set()
```

Repeats the evaluation of the data currently in the capture buffer without capturing new data. This is useful for long capture buffers that are split into sections for result displays. In this case, only the data in the currently selected capture buffer section is automatically refreshed after configuration changes. To update the entire capture buffer, you must refresh the evaluation manually using this command. See also ‘Capture buffer display’.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:REFMeas
driver.applications.k70Vsa.initiate.refMeas.set_with_opc()
```

Repeats the evaluation of the data currently in the capture buffer without capturing new data. This is useful for long capture buffers that are split into sections for result displays. In this case, only the data in the currently selected capture buffer section is automatically refreshed after configuration changes. To update the entire capture buffer, you must refresh the evaluation manually using this command. See also ‘Capture buffer display’.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.4.5 Refresh

SCPI Commands

```
INITiate:REFResh
```

class RefreshCls

Refresh commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:REFResh
driver.applications.k70Vsa.initiate.refresh.set()
```

This command updates the current measurement results to reflect the current measurement settings. No new I/Q data is captured. Thus, measurement settings apply to the I/Q data currently in the capture buffer. The command applies exclusively to I/Q measurements. It requires I/Q data.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:REFresh
driver.applications.k70Vsa.initiate.refresh.set_with_opc()
```

This command updates the current measurement results to reflect the current measurement settings. No new I/Q data is captured. Thus, measurement settings apply to the I/Q data currently in the capture buffer. The command applies exclusively to I/Q measurements. It requires I/Q data.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.5 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.applications.k70Vsa.inputPy.repcap_inputIx_get()
driver.applications.k70Vsa.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 8 total commands, 6 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.clone()
```

Subgroups

6.1.7.5.1 Attenuation

class AttenuationCls

Attenuation commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.attenuation.clone()
```

Subgroups

6.1.7.5.1.1 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.attenuation.auto.clone()
```

Subgroups

6.1.7.5.1.2 Mode

SCPI Commands

```
INPut<InputIx>:ATTenuation:AUTO:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → AttenuatorMode

```
# SCPI: INPut<ip>:ATTenuation:AUTO:MODE
value: enums.AttenuatorMode = driver.applications.k70Vsa.inputPy.attenuation.
↳ auto.mode.get(inputIx = repcap.InputIx.Default)
```

Selects the priority for signal processing after the RF attenuation has been applied.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

att_mode: No help available

set(att_mode: AttenuatorMode, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:ATTenuation:AUTO:MODE
driver.applications.k70Vsa.inputPy.attenuation.auto.mode.set(att_mode = enums.
↳ AttenuatorMode.LDISTortion, inputIx = repcap.InputIx.Default)
```

Selects the priority for signal processing after the RF attenuation has been applied.

param att_mode

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.7.5.2 Coupling

SCPI Commands

INPut<InputIx>:COUPling

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → CouplingTypeA

```
# SCPI: INPut<ip>:COUPling
value: enums.CouplingTypeA = driver.applications.k70Vsa.inputPy.coupling.
↳get(inputIx = repcap.InputIx.Default)
```

This command selects the coupling type of the RF input.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

input_coupling: No help available

set(input_coupling: CouplingTypeA, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:COUPling
driver.applications.k70Vsa.inputPy.coupling.set(input_coupling = enums.
↳CouplingTypeA.AC, inputIx = repcap.InputIx.Default)
```

This command selects the coupling type of the RF input.

param input_coupling

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.7.5.3 Dpath

SCPI Commands

INPut<InputIx>:DPATH

class DpathCls

Dpath commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → State

```
# SCPI: INPut<ip>:DPATH
value: enums.State = driver.applications.k70Vsa.inputPy.dpath.get(inputIx =
↳repcap.InputIx.Default)
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

auto_off: No help available

set(auto_off: State, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:DPATH
driver.applications.k70Vsa.inputPy.dpath.set(auto_off = enums.State.ALL,
↪inputIx = repcap.InputIx.Default)
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

param auto_off

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.7.5.4 FilterPy**class FilterPyCls**

FilterPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.filterPy.clone()
```

Subgroups**6.1.7.5.4.1 Hpass****class HpassCls**

Hpass commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.filterPy.hpass.clone()
```

Subgroups

6.1.7.5.4.2 State

SCPI Commands

```
INPut<InputIx>:FILTer:HPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:FILTer:HPASs[:STATe]
value: bool = driver.applications.k70Vsa.inputPy.filterPy.hpass.state.
↳ get(inputIx = repcap.InputIx.Default)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:FILTer:HPASs[:STATe]
driver.applications.k70Vsa.inputPy.filterPy.hpass.state.set(state = False,
↳ inputIx = repcap.InputIx.Default)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.7.5.4.3 Yig

class YigCls

Yig commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.filterPy.yig.clone()
```

Subgroups

6.1.7.5.4.4 State

SCPI Commands

```
INPut<InputIx>:FILTer:YIG:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:FILTer:YIG[:STATe]
value: bool = driver.applications.k70Vsa.inputPy.filterPy.yig.state.get(inputIx,
↪= repcap.InputIx.Default)
```

Enables or disables the YIG filter.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: ON | OFF | 0 | 1

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:FILTer:YIG[:STATe]
driver.applications.k70Vsa.inputPy.filterPy.yig.state.set(state = False,
↪inputIx = repcap.InputIx.Default)
```

Enables or disables the YIG filter.

param state

ON | OFF | 0 | 1

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.1.7.5.5 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.inputPy.gain.clone()
```

Subgroups

6.1.7.5.5.1 State

SCPI Commands

```
INPut<InputIx>:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:GAIN:STATe
value: bool = driver.applications.k70Vsa.inputPy.gain.state.get(inputIx = ↵
↵repcap.InputIx.Default)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:GAIN:STATe
driver.applications.k70Vsa.inputPy.gain.state.set(state = False, inputIx = ↵
↵repcap.InputIx.Default)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.7.5.5.2 Value**SCPI Commands**

INPut<InputIx>:GAIN:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

<pre># SCPI: INPut<ip>:GAIN[:VALue] value: float = driver.applications.k70Vsa.inputPy.gain.value.get(inputIx = ↵ ↵repcap.InputIx.Default)</pre>

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications. K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

<pre># SCPI: INPut<ip>:GAIN[:VALue] driver.applications.k70Vsa.inputPy.gain.value.set(value = 1.0, inputIx = repcap. ↵InputIx.Default)</pre>
--

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications. K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.7.5.6 Select

SCPI Commands

`INPut<InputIx>:SElect`

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → BbInputSource

```
# SCPI: INPut<ip>:SElect
value: enums.BbInputSource = driver.applications.k70Vsa.inputPy.select.
↳get(inputIx = repcap.InputIx.Default)
```

This command selects the signal source for measurements, i.e. it defines which connector is used to input data to the R&S FSWP. Tip: The I/Q data to be analyzed for VSA cannot only be measured by the R&S FSWP VSA application itself, it can also be imported to the application, provided it has the correct format. Furthermore, the analyzed I/Q data from the R&S FSWP VSA application can be exported for further analysis in external applications. For details, see the R&S FSWP I/Q Analyzer and I/Q Input User Manual.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

baseband_input_source: No help available

set(*baseband_input_source: BbInputSource, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:SElect
driver.applications.k70Vsa.inputPy.select.set(baseband_input_source = enums.
↳BbInputSource.AIQ, inputIx = repcap.InputIx.Default)
```

This command selects the signal source for measurements, i.e. it defines which connector is used to input data to the R&S FSWP. Tip: The I/Q data to be analyzed for VSA cannot only be measured by the R&S FSWP VSA application itself, it can also be imported to the application, provided it has the correct format. Furthermore, the analyzed I/Q data from the R&S FSWP VSA application can be exported for further analysis in external applications. For details, see the R&S FSWP I/Q Analyzer and I/Q Input User Manual.

param baseband_input_source

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.1.7.6 Layout

class LayoutCls

Layout commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.clone()
```

Subgroups

6.1.7.6.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.add.clone()
```

Subgroups

6.1.7.6.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeK70) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.applications.k70Vsa.layout.add.window.get(window_name = '1',
↪ direction = enums.WindowDirection.ABOVE, window_type = enums.WindowTypeK70.
↪ CaptureBufferMagnAbs=CBUFFER)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method RsFswp.Layout.Replace.Window.set command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_query**.

param direction

LEFT | RIGHT | ABOVE | BELOW Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.1.7.6.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.catalog.clone()
```

Subgroups

6.1.7.6.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.applications.k70Vsa.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return

result: No help available

6.1.7.6.3 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.identify.clone()
```

Subgroups

6.1.7.6.3.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.applications.k70Vsa.layout.identify.window.get(window_name,
↪= '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name

String containing the name of a window.

return

window_index: Index number of the window.

6.1.7.6.4 Move

class MoveCls

Move commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.move.clone()
```

Subgroups

6.1.7.6.4.1 Window

SCPI Commands

```
LAYout:MOVE:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(source_window: str, target_window: str, arg_2: WindowDirReplace) → None

```
# SCPI: LAYout:MOVE[:WINDow]
driver.applications.k70Vsa.layout.move.window.set(source_window = '1', target_
↪window = '1', arg_2 = enums.WindowDirReplace.ABOVe)
```

No command help available

param source_window

No help available

param target_window

No help available

param arg_2

No help available

6.1.7.6.5 Remove

class RemoveCls

Remove commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.remove.clone()
```

Subgroups

6.1.7.6.5.1 Window

SCPI Commands

```
LAYout:REMove:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(name: str) → None

```
# SCPI: LAYout:REMove[:WINDow]
driver.applications.k70Vsa.layout.remove.window.set(name = '1')
```

This command removes a window from the display in the active channel.

param name

String containing the name of the window. In the default state, the name of the window is its index.

6.1.7.6.6 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.layout.replace.clone()
```

Subgroups

6.1.7.6.6.1 Window

SCPI Commands

```
LAYout:REPLace:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeK70) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.applications.k70Vsa.layout.replace.window.set(window_name = '1', window_
↪type = enums.WindowTypeK70.CaptureBufferMagnAbs=CBuffer)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method `RsFswp.Layout.Add.Window.get_` command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method `RsFswp.Layout.Catalog.Window.get_` query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method `RsFswp.Layout.Add.Window.get_` for a list of available window types.

6.1.7.6.7 Splitter

SCPI Commands

```
LAYout:SPLitter
```

class SplitterCls

Splitter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*index_1*: int, *index_2*: int, *position*: int) → None

```
# SCPI: LAYout:SPLitter
driver.applications.k70Vsa.layout.splitter.set(index_1 = 1, index_2 = 1,
↪position = 1)
```

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method `RsFswp.Applications.K30_NoiseFigure.Display.Window.Size.set` command, the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

param index_1

The index of one window the splitter controls.

param index_2

The index of a window on the other side of the splitter.

param position

New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu). The point of origin (x = 0, y = 0) is in the lower left corner of the screen. The end point (x = 100, y = 100) is in the upper right corner of the screen. (See Figure ‘SmartGrid coordinates for remote control of the splitters’.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

6.1.7.7 MassMemory

class MassMemoryCls

MassMemory commands group definition. 9 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.clone()
```

Subgroups

6.1.7.7.1 Load<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.massMemory.load.repcap_window_get()
driver.applications.k70Vsa.massMemory.load.repcap_window_set(repcap.Window.Nr1)
```

class LoadCls

Load commands group definition. 4 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.load.clone()
```

Subgroups

6.1.7.7.1.1 Iq

class IqCls

Iq commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.load.iq.clone()
```

Subgroups

6.1.7.7.1.2 State

SCPI Commands

`MMEMory:LOAD<Window>:IQ:STaTe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*position: float, filename: str, window=Window.Default*) → None

```
# SCPI: MMEMory:LOAD<n>:IQ:STaTe
driver.applications.k70Vsa.massMemory.load.iq.state.set(position = 1.0,
↪filename = '1', window = repcap.Window.Default)
```

This command restores I/Q data from a file. The file extension is *.iq.tar.

param position

No help available

param filename

string String containing the path and name of the source file.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Load')

6.1.7.7.1.3 Stream

SCPI Commands

`MMEMory:LOAD:IQ:STReam`

class StreamCls

Stream commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:LOAD:IQ:STReam
value: str = driver.applications.k70Vsa.massMemory.load.iq.stream.get()
```

No command help available

return

channel: No help available

set(*channel: str*) → None

```
# SCPI: MMEMory:LOAD:IQ:STReam
driver.applications.k70Vsa.massMemory.load.iq.stream.set(channel = '1')
```

No command help available

param channel
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.load.iq.stream.clone()
```

Subgroups

6.1.7.7.1.4 Auto

SCPI Commands

```
MMEMory:LOAD:IQ:STReam:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:LOAD:IQ:STReam:AUTO
value: bool = driver.applications.k70Vsa.massMemory.load.iq.stream.auto.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:LOAD:IQ:STReam:AUTO
driver.applications.k70Vsa.massMemory.load.iq.stream.auto.set(state = False)
```

No command help available

param state
No help available

6.1.7.7.1.5 ListPy

SCPI Commands

```
MMEMory:LOAD:IQ:STReam:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:LOAD:IQ:STReam:LIST
value: str = driver.applications.k70Vsa.massMemory.load.iq.stream.listPy.get()
```

No command help available

```
return
channel: No help available
```

6.1.7.7.2 Store<Store>

RepCap Settings

```
# Range: Pos1 .. Pos32
rc = driver.applications.k70Vsa.massMemory.store.repcap_store_get()
driver.applications.k70Vsa.massMemory.store.repcap_store_set(repcap.Store.Pos1)
```

class StoreCls

Store commands group definition. 5 total commands, 2 Subgroups, 0 group commands Repeated Capability:
Store, default value after init: Store.Pos1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.store.clone()
```

Subgroups

6.1.7.7.2.1 Iq

class IqCls

Iq commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.massMemory.store.iq.clone()
```

Subgroups

6.1.7.7.2.2 Comment

SCPI Commands

```
MMEemory:STORe<Store>:IQ:COMMENT
```


class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → str

```
# SCPI: MMEMory:STORe<n>:IQ:COMMeNt
value: str = driver.applications.k70Vsa.massMemory.store.iq.comment.get(store = ↵
↵repcap.Store.Default)
```

This command adds a comment to a file that contains I/Q data.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

comment: String containing the comment.

set(comment: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:IQ:COMMeNt
driver.applications.k70Vsa.massMemory.store.iq.comment.set(comment = '1', store ↵
↵= repcap.Store.Default)
```

This command adds a comment to a file that contains I/Q data.

param comment

String containing the comment.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.1.7.7.2.3 FormatPy**SCPI Commands**

```
MMEMory:STORe<Store>:IQ:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → IqDataFormatDdem

```
# SCPI: MMEMory:STORe<n>:IQ:FORMat
value: enums.IqDataFormatDdem = driver.applications.k70Vsa.massMemory.store.iq.
↵formatPy.get(store = repcap.Store.Default)
```

This command sets or queries the format of the I/Q data to be stored.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

data_type: No help available

set(data_type: IqDataFormatDdem, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:IQ:FORMat
driver.applications.k70Vsa.massMemory.store.iq.formatPy.set(data_type = enums.
↪ IqDataFormatDdem.FloatComplex=FL0at32,COMPlEx, store = repcap.Store.Default)
```

This command sets or queries the format of the I/Q data to be stored.

param data_type

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.1.7.7.2.4 State

SCPI Commands

```
MMEemory:STORe<Store>:IQ:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(position: float, filename: str, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:IQ:STATe
driver.applications.k70Vsa.massMemory.store.iq.state.set(position = 1.0, ↪
↪ filename = '1', store = repcap.Store.Default)
```

This command writes the captured I/Q data to a file. The file extension is *.iq.tar. By default, the contents of the file are in 32-bit floating point format.

param position

1..n

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.1.7.7.2.5 Trace

SCPI Commands

```
MMEemory:STORe<Store>:IQ:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*position: float, filename: str, store=Store.Default*) → None

```
# SCPI: MMEemory:STORe<n>:IQ:TRACe
driver.applications.k70Vsa.massMemory.store.iq.trace.set(position = 1.0,
↪ filename = '1', store = repcap.Store.Default)
```

Stores the I/Q data for the displayed trace in the selected window to a file in iq.tar format. This command is only available for result types that provide I/Q data based on the error vector, such as the ‘Vector I/Q’ or Real/Imag displays.

param position

1..n Window

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.1.7.7.2.6 Trace

SCPI Commands

```
MMEemory:STORe<Store>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*trace: float, path: str, store=Store.Default*) → None

```
# SCPI: MMEemory:STORe<n>:TRACe
driver.applications.k70Vsa.massMemory.store.trace.set(trace = 1.0, path = '1',
↪ store = repcap.Store.Default)
```

This command exports trace data from the specified window to an ASCII file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a ‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device.

param trace

Number of the trace to be stored

param path

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.1.7.8 Output<OutputConnector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.applications.k70Vsa.output.repcap_outputConnector_get()
driver.applications.k70Vsa.output.repcap_outputConnector_set(repcap.OutputConnector.Nr1)
```

class OutputCls

Output commands group definition. 6 total commands, 2 Subgroups, 0 group commands Repeated Capability: OutputConnector, default value after init: OutputConnector.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.output.clone()
```

Subgroups

6.1.7.8.1 Ifreq

class IfreqCls

Ifreq commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.output.ifreq.clone()
```

Subgroups

6.1.7.8.1.1 Source

SCPI Commands

```
OUTPut<OutputConnector>:IF:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → Source

```
# SCPI: OUTPut<up>:IF[:SOURce]
value: enums.Source = driver.applications.k70Vsa.output.ifreq.source.
↳ get(outputConnector = repcap.OutputConnector.Default)
```

Defines the type of signal available at one of the output connectors of the R&S FSWP.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

source: IF The measured IF value is available at the IF/VIDEO/DEMODO output connector. VIDEO The displayed video signal (i.e. the filtered and detected IF signal, 200mV) is available at the IF/VIDEO/DEMODO output connector. This setting is required to provide demodulated audio frequencies at the output.

set(source: Source, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:IF[:SOURce]
driver.applications.k70Vsa.output.ifreq.source.set(source = enums.Source.AM,
↪outputConnector = repcap.OutputConnector.Default)
```

Defines the type of signal available at one of the output connectors of the R&S FSWP.

param source

IF The measured IF value is available at the IF/VIDEO/DEMODO output connector. VIDEO The displayed video signal (i.e. the filtered and detected IF signal, 200mV) is available at the IF/VIDEO/DEMODO output connector. This setting is required to provide demodulated audio frequencies at the output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

6.1.7.8.2 Trigger<TriggerPort>**RepCap Settings**

```
# Range: Nr1 .. Nr8
rc = driver.applications.k70Vsa.output.trigger.repcap_triggerPort_get()
driver.applications.k70Vsa.output.trigger.repcap_triggerPort_set(repcap.TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 4 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.output.trigger.clone()
```

Subgroups

6.1.7.8.2.1 Direction

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → InOutDirection

```
# SCPI: OUTPut<up>:TRIGger<tp>:DIRection
value: enums.InOutDirection = driver.applications.k70Vsa.output.trigger.
↳ direction.get(outputConnector = repcap.OutputConnector.Default, triggerPort =
↳ repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

trigger_output_direction: No help available

set(trigger_output_direction: InOutDirection, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:DIRection
driver.applications.k70Vsa.output.trigger.direction.set(trigger_output_
↳ direction = enums.InOutDirection.INPUT, outputConnector = repcap.
↳ OutputConnector.Default, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param trigger_output_direction

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.8.2.2 Level

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → LowHigh

```
# SCPI: OUTPut<up>:TRIGger<tp>:LEVel
value: enums.LowHigh = driver.applications.k70Vsa.output.trigger.level.
↳get(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↳TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

trigger_output_level: No help available

set(*trigger_output_level: LowHigh, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:LEVel
driver.applications.k70Vsa.output.trigger.level.set(trigger_output_level =
↳enums.LowHigh.HIGH, outputConnector = repcap.OutputConnector.Default,
↳triggerPort = repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param trigger_output_level

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.8.2.3 Otype

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<tp>:OTYPE
```

class OtypeCls

Otype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → OutputType

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPE
value: enums.OutputType = driver.applications.k70Vsa.output.trigger.otype.
↳get(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↳TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

trigger_output_type: No help available

set(trigger_output_type: OutputType, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPE
driver.applications.k70Vsa.output.trigger.otype.set(trigger_output_type = enums.
↳OutputType.DEVICE, outputConnector = repcap.OutputConnector.Default,
↳triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output.

param trigger_output_type

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.8.2.4 Pulse

class PulseCls

Pulse commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.output.trigger.pulse.clone()
```

Subgroups

6.1.7.8.2.5 Immediate

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:IMMediate
driver.applications.k70Vsa.output.trigger.pulse.immediate.set(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.TriggerPort.Default)
```

This command generates a pulse at the trigger output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

set_with_opc(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default, opc_timeout_ms: int = -1) → None

6.1.7.8.2.6 Length

SCPI Commands

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → float

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:LENGth
value: float = driver.applications.k70Vsa.output.trigger.pulse.length.
↪get(outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↪TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

trigger_output_length: No help available

set(*trigger_output_length: float, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default*) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:PULSe:LENGth
driver.applications.k70Vsa.output.trigger.pulse.length.set(trigger_output_
↪length = 1.0, outputConnector = repcap.OutputConnector.Default, triggerPort =
↪recap.TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param trigger_output_length

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.9 Sense

class SenseCls

Sense commands group definition. 142 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.clone()
```

Subgroups

6.1.7.9.1 Adjust

class AdjustCls

Adjust commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.adjust.clone()
```

Subgroups

6.1.7.9.1.1 Configure

class ConfigureCls

Configure commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.adjust.configure.clone()
```

Subgroups

6.1.7.9.1.2 Hysteresis

class HysteresisCls

Hysteresis commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.adjust.configure.hysteresis.clone()
```

Subgroups

6.1.7.9.1.3 Lower

SCPI Commands

```
SENSe:ADJust:CONFigure:HYSTeresis:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure[:HYSTeresis]:LOWer
value: float = driver.applications.k70Vsa.sense.adjust.configure.hysteresis.
↳ lower.get()
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines a lower threshold the signal must fall below (compared to the last measurement) before the reference level is adapted automatically.

return

hysteresis_lower: No help available

set(hysteresis_lower: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure[:HYSTeresis]:LOWer
driver.applications.k70Vsa.sense.adjust.configure.hysteresis.lower.
↳ set(hysteresis_lower = 1.0)
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines a lower threshold the signal must fall below (compared to the last measurement) before the reference level is adapted automatically.

param hysteresis_lower

Range: 0 dB to 200 dB, Unit: dB

6.1.7.9.1.4 Upper**SCPI Commands**

```
SENSe:ADJust:CONFigure:HYSTeresis:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure[:HYSTeresis]:UPPer
value: float = driver.applications.k70Vsa.sense.adjust.configure.hysteresis.
↳ upper.get()
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines an upper threshold the signal must exceed (compared to the last measurement) before the reference level is adapted automatically.

return

hysteresis_upper: No help available

`set(hysteresis_upper: float) → None`

```
# SCPI: [SENSe]:ADJust:CONFigure[:HYSTeresis]:UPPer
driver.applications.k70Vsa.sense.adjust.configure.hysteresis.upper.
↪set(hysteresis_upper = 1.0)
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines an upper threshold the signal must exceed (compared to the last measurement) before the reference level is adapted automatically.

param hysteresis_upper
Range: 0 dB to 200 dB, Unit: dB

6.1.7.9.1.5 Level

class LevelCls

Level commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.adjust.configure.level.clone()
```

Subgroups

6.1.7.9.1.6 Duration

class DurationCls

Duration commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.adjust.configure.level.duration.clone()
```

Subgroups

6.1.7.9.1.7 Mode

SCPI Commands

```
SENSe:ADJust:CONFigure:LEVel:DURation:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:ADJust:CONFigure[:LEVel]:DURation:MODE
value: enums.AutoManualMode = driver.applications.k70Vsa.sense.adjust.configure.
↳level.duration.mode.get()
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command selects the way the R&S FSWP determines the length of the measurement .

return

auto_auto_level: No help available

set(auto_auto_level: AutoManualMode) → None

```
# SCPI: [SENSe]:ADJust:CONFigure[:LEVel]:DURation:MODE
driver.applications.k70Vsa.sense.adjust.configure.level.duration.mode.set(auto_
↳auto_level = enums.AutoManualMode.AUTO)
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command selects the way the R&S FSWP determines the length of the measurement .

param auto_auto_level

AUTO The R&S FSWP determines the measurement length automatically according to the current input data. MANual The R&S FSWP uses the measurement length defined by [SENSe:]ADJust:CONFigure:LEVel:DURation.

6.1.7.9.1.8 Level

SCPI Commands

SENSe:ADJust:LEVel

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k70Vsa.sense.adjust.level.set()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.applications.k70Vsa.sense.adjust.level.set_with_opc()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.9.2 Ddemod

class DdemodCls

Ddemod commands group definition. 131 total commands, 32 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.clone()
```

Subgroups

6.1.7.9.2.1 Apsk

class ApskCls

Apsk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.apsk.clone()
```

Subgroups

6.1.7.9.2.2 Nstate

SCPI Commands

```
SENSe:DDEMod:APSK:NState
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:APSK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.apsk.nstate.get()
```

This command defines the specific demodulation mode for APSK.

return

apskn_state: 16 | 32 16 16APSK 32 32APSK

set(*apskn_state: float*) → None

```
# SCPI: [SENSe]:DDEMod:APSK:NState
driver.applications.k70Vsa.sense.ddemod.apsk.nstate.set(apskn_state = 1.0)
```

This command defines the specific demodulation mode for APSK.

```
param apskn_state
    16 | 32 16 16APSK 32 32APSK
```

6.1.7.9.2.3 Ask

class AskCls

Ask commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.ask.clone()
```

Subgroups

6.1.7.9.2.4 Nstate

SCPI Commands

```
SENSe:DDEMod:ASK:NState
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:ASK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.ask.nstate.get()
```

This command defines the specific demodulation mode for ASK.

```
return
    askn_state: 2 | 4 2 OOK 4 4ASK
```

set(*askn_state: float*) → None

```
# SCPI: [SENSe]:DDEMod:ASK:NState
driver.applications.k70Vsa.sense.ddemod.ask.nstate.set(askn_state = 1.0)
```

This command defines the specific demodulation mode for ASK.

```
param askn_state
    2 | 4 2 OOK 4 4ASK
```


6.1.7.9.2.5 Bordering

SCPI Commands

```
SENSe:DDEMod:BORDERing
```

class BorderingCls

Bordering commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → BitOrdering

```
# SCPI: [SENSe]:DDEMod:BORDERing
value: enums.BitOrdering = driver.applications.k70Vsa.sense.ddemod.bordering.
↳get()
```

Determines how the bits in the symbols are ordered in all symbol displays.

return

bit_ordering: MSB | LSB LSB Least-significant bit first (used in Bluetooth specification, for example) MSB Most significant bit first (default)

set(bit_ordering: BitOrdering) → None

```
# SCPI: [SENSe]:DDEMod:BORDERing
driver.applications.k70Vsa.sense.ddemod.bordering.set(bit_ordering = enums.
↳BitOrdering.LSB)
```

Determines how the bits in the symbols are ordered in all symbol displays.

param bit_ordering

MSB | LSB LSB Least-significant bit first (used in Bluetooth specification, for example) MSB Most significant bit first (default)

6.1.7.9.2.6 Ecalc

class EcalcCls

Ecalc commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.ecalc.clone()
```

Subgroups

6.1.7.9.2.7 Mode

SCPI Commands

SENSe:DDemod:ECALc:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EvmCalc

```
# SCPI: [SENSe]:DDEMod:ECALc[:MODE]
value: enums.EvmCalc = driver.applications.k70Vsa.sense.ddemod.ecalc.mode.get()
```

This command defines the calculation formula for EVM.

return

evm_calc: SIGNAL | SYMBOL | MECPower | MACPower SIGNAL Calculation normalized to the mean power of the reference signal at the symbol instants. SYMBOL Calculation normalized to the maximum power of the reference signal at the symbol instants. MECPower Calculation normalized to the mean expected power of the measurement signal at the symbol instants MACPower Calculation normalized to the maximum expected power of the measurement signal at the symbol instants

set(evm_calc: EvmCalc) → None

```
# SCPI: [SENSe]:DDEMod:ECALc[:MODE]
driver.applications.k70Vsa.sense.ddemod.ecalc.mode.set(evm_calc = enums.EvmCalc.
↳MACPower)
```

This command defines the calculation formula for EVM.

param evm_calc

SIGNAL | SYMBOL | MECPower | MACPower SIGNAL Calculation normalized to the mean power of the reference signal at the symbol instants. SYMBOL Calculation normalized to the maximum power of the reference signal at the symbol instants. MECPower Calculation normalized to the mean expected power of the measurement signal at the symbol instants MACPower Calculation normalized to the maximum expected power of the measurement signal at the symbol instants

6.1.7.9.2.8 Offset

SCPI Commands

SENSe:DDemod:ECALc:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:ECALc:OFFSet
value: bool = driver.applications.k70Vsa.sense.ddemod.ecalc.offset.get()
```

Configures the way the VSA application calculates the error vector results for offset QPSK.

return

state: ON | 1 VSA application compensates the delay of the Q component with respect to the I component in the measurement signal as well as the reference signal before calculating the error vector. That means that the error vector contains only one symbol instant per symbol period. OFF | 0 The VSA application subtracts the measured signal from the reference signal to calculate the error vector. This method results in the fact that the error vector contains two symbol instants per symbol period: one that corresponds to the I component and one that corresponds to the Q component.

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:ECALc:OFFSet
driver.applications.k70Vsa.sense.ddemod.ecalc.offset.set(state = False)
```

Configures the way the VSA application calculates the error vector results for offset QPSK.

param state

ON | 1 VSA application compensates the delay of the Q component with respect to the I component in the measurement signal as well as the reference signal before calculating the error vector. That means that the error vector contains only one symbol instant per symbol period. OFF | 0 The VSA application subtracts the measured signal from the reference signal to calculate the error vector. This method results in the fact that the error vector contains two symbol instants per symbol period: one that corresponds to the I component and one that corresponds to the Q component.

6.1.7.9.2.9 EpRate

class EpRateCls

EpRate commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.epRate.clone()
```

Subgroups

6.1.7.9.2.10 Auto

SCPI Commands

```
SENSe:DDEMod:EPRate:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:EPRate:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.epRate.auto.get()
```

Defines how many sample points are used at each symbol to calculate modulation accuracy results automatically. If enabled, the VSA application uses the following settings, depending on the modulation type:

Table Header: Modulation / Est. Points

- PSK, QAM / 1
- Offset QPSK / 2
- FSK, MSK / Sample rate (see [SENSe:]DDEMod:PRATe)

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:EPRate:AUTO
driver.applications.k70Vsa.sense.ddemod.epRate.auto.set(state = False)
```

Defines how many sample points are used at each symbol to calculate modulation accuracy results automatically. If enabled, the VSA application uses the following settings, depending on the modulation type:

Table Header: Modulation / Est. Points

- PSK, QAM / 1
- Offset QPSK / 2
- FSK, MSK / Sample rate (see [SENSe:]DDEMod:PRATe)

param state

No help available

6.1.7.9.2.11 Value

SCPI Commands

```
SENSe:DDEMod:EPRate:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:EPRate[:VALue]
value: float = driver.applications.k70Vsa.sense.ddemod.epRate.value.get()
```

Defines how many sample points are used at each symbol to calculate modulation accuracy results. For more information see ‘Estimation points per symbol’. You can also let the VSA application decide how many estimation points to use, see [SENSe:]DDEMod:EPRate:AUTO.

return

est_oversampling: 1 the estimation algorithm takes only the symbol time instants into account 2 two points per symbol instant are used (required for Offset QPSK) 4 | 8 | 16 | 32 the number of samples per symbol defined in the signal capture settings is used (see [SENSe:]DDEMod:PRATe) , i.e. all sample time instants are weighted equally

set(est_oversampling: float) → None

```
# SCPI: [SENSe]:DDEMod:EPRate[:VALue]
driver.applications.k70Vsa.sense.ddemod.epRate.value.set(est_oversampling = 1.0)
```

Defines how many sample points are used at each symbol to calculate modulation accuracy results. For more information see ‘Estimation points per symbol’. You can also let the VSA application decide how many estimation points to use, see [SENSe:]DDEMod:EPRate:AUTO.

param est_oversampling

1 the estimation algorithm takes only the symbol time instants into account 2 two points per symbol instant are used (required for Offset QPSK) 4 | 8 | 16 | 32 the number of samples per symbol defined in the signal capture settings is used (see [SENSe:]DDEMod:PRATe) , i.e. all sample time instants are weighted equally

6.1.7.9.2.12 Equalizer**SCPI Commands**

```
SENSe:DDEMod:EQualizer:RESet
```

class EqualizerCls

Equalizer commands group definition. 7 total commands, 6 Subgroups, 1 group commands

reset() → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:RESet
driver.applications.k70Vsa.sense.ddemod.equalizer.reset()
```

This command deletes the data of the currently selected equalizer. After deletion, training can start again using the command DDEM:EQU:MODE TRA (see [SENSe:]DDEMod:EQualizer:MODE) .

reset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:RESet
driver.applications.k70Vsa.sense.ddemod.equalizer.reset_with_opc()
```

This command deletes the data of the currently selected equalizer. After deletion, training can start again using the command DDEM:EQU:MODE TRA (see [SENSe:]DDEMod:EQualizer:MODE) .

Same as reset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.equalizer.clone()
```

Subgroups

6.1.7.9.2.13 File

class FileCls

File commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.equalizer.file.clone()
```

Subgroups

6.1.7.9.2.14 FormatPy

SCPI Commands

```
SENSe:DDEMod:EQualizer:FILE:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FileFormatDdem

```
# SCPI: [SENSe]:DDEMod:EQualizer:FILE:FORMat
value: enums.FileFormatDdem = driver.applications.k70Vsa.sense.ddemod.equalizer.
↳ file.formatPy.get()
```

Determines the file format for stored equalizer results.

return

eq_format: VAE | FRES VAE To be used as an equalizer file in VSA applications FRES
To be used as a user-defined frequency response correction file in any other application
that supports it

set(eq_format: FileFormatDdem) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:FILE:FORMat
driver.applications.k70Vsa.sense.ddemod.equalizer.file.formatPy.set(eq_format =
↳ enums.FileFormatDdem.FRES)
```

Determines the file format for stored equalizer results.

param eq_format

VAE | FRES VAE To be used as an equalizer file in VSA applications FRES To be used as a user-defined frequency response correction file in any other application that supports it

6.1.7.9.2.15 Length**SCPI Commands**

```
SENSe:DDMod:EQualizer:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDMod:EQualizer:LENGth
value: float = driver.applications.k70Vsa.sense.ddemod.equalizer.length.get()
```

This command defines the length of the equalizer in terms of symbols.

return

length: Range: 1 to 256, Unit: SYMB

set(length: float) → None

```
# SCPI: [SENSe]:DDMod:EQualizer:LENGth
driver.applications.k70Vsa.sense.ddemod.equalizer.length.set(length = 1.0)
```

This command defines the length of the equalizer in terms of symbols.

param length

Range: 1 to 256, Unit: SYMB

6.1.7.9.2.16 Load**SCPI Commands**

```
SENSe:DDMod:EQualizer:LOAD
```

class LoadCls

Load commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDMod:EQualizer:LOAD
value: str = driver.applications.k70Vsa.sense.ddemod.equalizer.load.get()
```

This command selects a user-defined equalizer. The equalizer mode is automatically switched to USER (see [SENSe:]DDMod:EQualizer:MODE) .

return

filename: Path and file name (without extension)

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:LOAD
driver.applications.k70Vsa.sense.ddemod.equalizer.load.set(filename = '1')
```

This command selects a user-defined equalizer. The equalizer mode is automatically switched to USER (see [SENSe:]DDEMod:EQualizer:MODE).

param filename

Path and file name (without extension)

6.1.7.9.2.17 Mode

SCPI Commands

```
SENSe:DDEMod:EQualizer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → IfGainModeDdem

```
# SCPI: [SENSe]:DDEMod:EQualizer:MODE
value: enums.IfGainModeDdem = driver.applications.k70Vsa.sense.ddemod.equalizer.
↳mode.get()
```

Switches between the equalizer modes. For details see ‘The equalizer’.

return

mode: NORMal Switches the equalizer on for the next sweep. TRACKing Switches the equalizer on; the results of the equalizer in the previous sweep are considered to calculate the new filter. FREeze The filter is no longer changed, the current equalizer values are used for subsequent sweeps. USER A user-defined equalizer loaded from a file is used. AVERaging Switches the equalizer on; the results of the equalizer in all previous sweeps (since the instrument was switched on or the equalizer was reset) are considered to calculate the new filter. To start a new averaging process, use the [SENSe:]DDEMod:EQualizer:RESet command.

set(mode: IfGainModeDdem) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:MODE
driver.applications.k70Vsa.sense.ddemod.equalizer.mode.set(mode = enums.
↳IfGainModeDdem.AVERaging)
```

Switches between the equalizer modes. For details see ‘The equalizer’.

param mode

NORMal Switches the equalizer on for the next sweep. TRACKing Switches the equalizer on; the results of the equalizer in the previous sweep are considered to calculate the new filter. FREeze The filter is no longer changed, the current equalizer values are used for subsequent sweeps. USER A user-defined equalizer loaded from a file is used. AVERaging Switches the equalizer on; the results of the equalizer in all previous sweeps (since the instrument was switched on or the equalizer was reset) are considered to calculate the new filter. To start a new averaging process, use the [SENSe:]DDEMod:EQualizer:RESet command.

6.1.7.9.2.18 Save

SCPI Commands

```
SENSe:DDEMod:EQualizer:SAVE
```

class SaveCls

Save commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:EQualizer:SAVE
value: str = driver.applications.k70Vsa.sense.ddemod.equalizer.save.get()
```

This command saves the current equalizer results to a file.

return
filename: File name

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer:SAVE
driver.applications.k70Vsa.sense.ddemod.equalizer.save.set(filename = '1')
```

This command saves the current equalizer results to a file.

param filename
File name

6.1.7.9.2.19 State

SCPI Commands

```
SENSe:DDEMod:EQualizer:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:EQualizer[:STATe]
value: bool = driver.applications.k70Vsa.sense.ddemod.equalizer.state.get()
```

This command activates or deactivates the equalizer. For more information on the equalizer see ‘The equalizer’.

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:EQualizer[:STATe]
driver.applications.k70Vsa.sense.ddemod.equalizer.state.set(state = False)
```

This command activates or deactivates the equalizer. For more information on the equalizer see ‘The equalizer’.

param state

No help available

6.1.7.9.2.20 Factory

class FactoryCls

Factory commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.factory.clone()
```

Subgroups

6.1.7.9.2.21 Value

SCPI Commands

```
SENSe:DDEMod:FACTory:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(factory: Factory) → None

```
# SCPI: [SENSe]:DDEMod:FACTory[:VALue]
driver.applications.k70Vsa.sense.ddemod.factory.value.set(factory = enums.
↳Factory.ALL)
```

This command restores the factory settings of standards or patterns for the VSA application.

param factory

ALL | STANdard | PATTern ALL Restores both standards and patterns.

6.1.7.9.2.22 FilterPy

class FilterPyCls

FilterPy commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.filterPy.clone()
```

Subgroups

6.1.7.9.2.23 Alpha

SCPI Commands

```
SENSe:DDEMod:FILTer:ALPHa
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:FILTer:ALPHa
value: float = driver.applications.k70Vsa.sense.ddemod.filterPy.alpha.get()
```

This command determines the filter characteristic (ALPHA/BT) .

```
return
    meas_filter_alpha_bt: Range: 0.03 to 1.0
```

set(meas_filter_alpha_bt: float) → None

```
# SCPI: [SENSe]:DDEMod:FILTer:ALPHa
driver.applications.k70Vsa.sense.ddemod.filterPy.alpha.set(meas_filter_alpha_bt,
↳ 1.0)
```

This command determines the filter characteristic (ALPHA/BT) .

```
param meas_filter_alpha_bt
    Range: 0.03 to 1.0
```

6.1.7.9.2.24 Reference

SCPI Commands

```
SENSe:DDEMod:FILTer:REFeRence
```

class ReferenceCls

Reference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DdemodFilter

```
# SCPI: [SENSe]:DDEMod:FILTer:REFeRence
value: enums.DdemodFilter = driver.applications.k70Vsa.sense.ddemod.filterPy.
↳ reference.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: *DdemodFilter*) → None

```
# SCPI: [SENSe]:DDEMod:FILTer:REference
driver.applications.k70Vsa.sense.ddemod.filterPy.reference.set(arg_0 = enums.
↪ DdemodFilter.A25Fm)
```

No command help available

```
param arg_0
    No help available
```

6.1.7.9.2.25 State

SCPI Commands

SENSe:DDEMod:FILTer:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:FILTer[:STATe]
value: bool = driver.applications.k70Vsa.sense.ddemod.filterPy.state.get()
```

This command defines whether the input signal that is evaluated is filtered by the measurement filter. This command has no effect on the transmit filter.

```
return
    state: ON | 1 [SENSe:]DDEMod:MFILter:AUTO is activated. OFF | 0 The input signal
        is not filtered. [SENSe:]DDEMod:MFILter:AUTO is deactivated.
```

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:FILTer[:STATe]
driver.applications.k70Vsa.sense.ddemod.filterPy.state.set(state = False)
```

This command defines whether the input signal that is evaluated is filtered by the measurement filter. This command has no effect on the transmit filter.

```
param state
    ON | 1 [SENSe:]DDEMod:MFILter:AUTO is activated. OFF | 0 The input signal is
        not filtered. [SENSe:]DDEMod:MFILter:AUTO is deactivated.
```

6.1.7.9.2.26 FormatPy

SCPI Commands

```
SENSe:DDEMod:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DdemGroup

```
# SCPI: [SENSe]:DDEMod:FORMat
value: enums.DdemGroup = driver.applications.k70Vsa.sense.ddemod.formatPy.get()
```

This command selects the digital demodulation mode.

return

group: MSK | PSK | QAM | QPSK | FSK | ASK | APSK | UQAM QPSK Quad Phase Shift Key PSK Phase Shift Key MSK Minimum Shift Key QAM Quadrature Amplitude Modulation FSK Frequency Shift Key ASK Amplitude Shift Keying APSK Amplitude Phase Shift Keying UQAM User-defined modulation (loaded from file, see [SENSe:]DDEMod:USER:NAME)

set(group: DdemGroup) → None

```
# SCPI: [SENSe]:DDEMod:FORMat
driver.applications.k70Vsa.sense.ddemod.formatPy.set(group = enums.DdemGroup.
    ↪APSK)
```

This command selects the digital demodulation mode.

param group

MSK | PSK | QAM | QPSK | FSK | ASK | APSK | UQAM QPSK Quad Phase Shift Key PSK Phase Shift Key MSK Minimum Shift Key QAM Quadrature Amplitude Modulation FSK Frequency Shift Key ASK Amplitude Shift Keying APSK Amplitude Phase Shift Keying UQAM User-defined modulation (loaded from file, see [SENSe:]DDEMod:USER:NAME)

6.1.7.9.2.27 Fsk

class FskCls

Fsk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.fsk.clone()
```

Subgroups

6.1.7.9.2.28 Nstate

SCPI Commands

SENSe:DEMod:FSK:NState

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DEMod:FSK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.fsk.nstate.get()
```

This command defines the demodulation of the FSK modulation scheme.

return

fskn_state: 2 | 4 | 8 | 16 2 2FSK 4 4FSK 8 8FSK 16 16FSK

set(fskn_state: float) → None

```
# SCPI: [SENSe]:DEMod:FSK:NState
driver.applications.k70Vsa.sense.ddemod.fsk.nstate.set(fskn_state = 1.0)
```

This command defines the demodulation of the FSK modulation scheme.

param fskn_state

2 | 4 | 8 | 16 2 2FSK 4 4FSK 8 8FSK 16 16FSK

6.1.7.9.2.29 Fsync

class FsyncCls

Fsync commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.fsync.clone()
```

Subgroups

6.1.7.9.2.30 Auto

SCPI Commands

SENSe:DEMod:FSYNc:AUTO

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:FSYNc:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.fsync.auto.get()
```

This command selects manual or automatic Fine Sync

return
 state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:FSYNc:AUTO
driver.applications.k70Vsa.sense.ddemod.fsync.auto.set(state = False)
```

This command selects manual or automatic Fine Sync

param state
 ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.31 Level**SCPI Commands**

```
SENSe:DDEMod:FSYNc:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:FSYNc:LEVel
value: float = driver.applications.k70Vsa.sense.ddemod.fsync.level.get()
```

This command sets the Fine Sync Level if fine sync works on Known Data

return
 ser_level: Range: 0.0 to 100.0, Unit: PCT

set(ser_level: float) → None

```
# SCPI: [SENSe]:DDEMod:FSYNc:LEVel
driver.applications.k70Vsa.sense.ddemod.fsync.level.set(ser_level = 1.0)
```

This command sets the Fine Sync Level if fine sync works on Known Data

param ser_level
 Range: 0.0 to 100.0, Unit: PCT

6.1.7.9.2.32 Mode

SCPI Commands

SENSe:DDEMod:FSYNc:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FineSync

```
# SCPI: [SENSe]:DDEMod:FSYNc[:MODE]
value: enums.FineSync = driver.applications.k70Vsa.sense.ddemod.fsync.mode.get()
```

This command defines the fine synchronization mode used to calculate results, e.g. the bit error rate. Note: You can define a maximum symbol error rate (SER) for the known data in reference to the analyzed data. If the SER of the known data exceeds this limit, the default synchronization using the detected data is performed. See [SENSe:]DDEMod:FSYNc:LEVel.

return

fine_sync: KDATA | PATTeRn | DDATA KDATA (Default) The reference signal is defined as the data sequence from the loaded Known Data file that most closely matches the measured data. PATTeRn The reference signal is estimated from the defined pattern. This setting requires an activated pattern search, see [SENSe:]DDEMod:SEARch:SYNc:STATe. DDATA The reference signal is estimated from the detected data.

set(fine_sync: FineSync) → None

```
# SCPI: [SENSe]:DDEMod:FSYNc[:MODE]
driver.applications.k70Vsa.sense.ddemod.fsync.mode.set(fine_sync = enums.
↳ FineSync.DDATA)
```

This command defines the fine synchronization mode used to calculate results, e.g. the bit error rate. Note: You can define a maximum symbol error rate (SER) for the known data in reference to the analyzed data. If the SER of the known data exceeds this limit, the default synchronization using the detected data is performed. See [SENSe:]DDEMod:FSYNc:LEVel.

param fine_sync

KDATA | PATTeRn | DDATA KDATA (Default) The reference signal is defined as the data sequence from the loaded Known Data file that most closely matches the measured data. PATTeRn The reference signal is estimated from the defined pattern. This setting requires an activated pattern search, see [SENSe:]DDEMod:SEARch:SYNc:STATe. DDATA The reference signal is estimated from the detected data.

6.1.7.9.2.33 Result

SCPI Commands

```
SENSe:DDemod:FSYNc:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDemod:FSYNc:RESult
value: bool = driver.applications.k70Vsa.sense.ddemod.fsync.result.get()
```

Queries the result of the fine sync.

return

result: ON | OFF | 0 | 1 OFF | 0 fine sync with known data failed ON | 1 fine sync with known data successful

6.1.7.9.2.34 Kdata

class KdataCls

Kdata commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.kdata.clone()
```

Subgroups

6.1.7.9.2.35 Name

SCPI Commands

```
SENSe:DDemod:KDATA:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDemod:KDATA[:NAME]
value: str = driver.applications.k70Vsa.sense.ddemod.kdata.name.get()
```

This command selects the Known Data file. Note that known data must be activated ([SENSe:]DDemod:KDATA:STATe) before you can select a file.

return

filename: No help available

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:KDATA[:NAME]
driver.applications.k70Vsa.sense.ddemod.kdata.name.set(filename = '1')
```

This command selects the Known Data file. Note that known data must be activated ([SENSe]:DDEMod:KDATA:STATe) before you can select a file.

param filename
No help available

6.1.7.9.2.36 Ser

class SerCls

Ser commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.kdata.ser.clone()
```

Subgroups

6.1.7.9.2.37 Limit

SCPI Commands

```
SENSe:DDEMod:KDATA:SER:LIMit
```

class LimitCls

Limit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:KDATA:SER:LIMit
value: bool = driver.applications.k70Vsa.sense.ddemod.kdata.ser.limit.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:KDATA:SER:LIMit
driver.applications.k70Vsa.sense.ddemod.kdata.ser.limit.set(state = False)
```

No command help available

param state
No help available

6.1.7.9.2.38 State

SCPI Commands

```
SENSe:DDEMod:KDATA:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:KDATA:STATe
value: bool = driver.applications.k70Vsa.sense.ddemod.kdata.state.get()
```

This command selects the Known Data state. The use of known data is a prerequisite for the BER measurement and can also be used for the fine sync.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:KDATA:STATe
driver.applications.k70Vsa.sense.ddemod.kdata.state.set(state = False)
```

This command selects the Known Data state. The use of known data is a prerequisite for the BER measurement and can also be used for the fine sync.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.39 Mapping

class MappingCls

Mapping commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.mapping.clone()
```

Subgroups

6.1.7.9.2.40 Catalog

SCPI Commands

```
SENSe:DDEMod:MAPPING:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:MAPPing:CATalog
value: str = driver.applications.k70Vsa.sense.ddemod.mapping.catalog.get()
```

This command queries the names of all mappings that are available for the current modulation type and order. A mapping describes the assignment of constellation points to symbols.

return

mappings: list A comma-separated list of strings, with one string for each mapping name.

6.1.7.9.2.41 Value**SCPI Commands**

```
SENSe:DDEMod:MAPPing:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:MAPPing[:VALue]
value: str = driver.applications.k70Vsa.sense.ddemod.mapping.value.get()
```

This command selects the mapping for digital demodulation. The mapping describes the assignment of constellation points to symbols.

return

mapping: To obtain a list of available symbol mappings for the current modulation type use the [SENSe:]DDEMod:MAPPing:CATalog?? query.

set(mapping: str) → None

```
# SCPI: [SENSe]:DDEMod:MAPPing[:VALue]
driver.applications.k70Vsa.sense.ddemod.mapping.value.set(mapping = '1')
```

This command selects the mapping for digital demodulation. The mapping describes the assignment of constellation points to symbols.

param mapping

To obtain a list of available symbol mappings for the current modulation type use the [SENSe:]DDEMod:MAPPing:CATalog?? query.

6.1.7.9.2.42 Mfilter

class MfilterCls

Mfilter commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.mfilter.clone()
```

Subgroups

6.1.7.9.2.43 Alpha

SCPI Commands

```
SENSe:DDEMod:MFILter:ALPHA
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:MFILter:ALPHA
value: float = driver.applications.k70Vsa.sense.ddemod.mfilter.alpha.get()
```

This command sets the alpha value of the measurement filter.

return

meas_filter_alpha_bt: Range: 0.03 to 1.0, Unit: none

set(meas_filter_alpha_bt: float) → None

```
# SCPI: [SENSe]:DDEMod:MFILter:ALPHA
driver.applications.k70Vsa.sense.ddemod.mfilter.alpha.set(meas_filter_alpha_bt,
↳ 1.0)
```

This command sets the alpha value of the measurement filter.

param meas_filter_alpha_bt

Range: 0.03 to 1.0, Unit: none

6.1.7.9.2.44 Auto

SCPI Commands

```
SENSe:DDEMod:MFILter:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:MFILter:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.mfilter.auto.get()
```

If this command is set to 'ON', the measurement filter is defined automatically depending on the transmit filter (see [SENSe]:DDEMod:TFILter:NAME) .

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:MFILter:AUTO
driver.applications.k70Vsa.sense.ddemod.mfilter.auto.set(state = False)
```

If this command is set to 'ON', the measurement filter is defined automatically depending on the transmit filter (see [SENSe]:DDEMod:TFILter:NAME) .

param state
No help available

6.1.7.9.2.45 Name

SCPI Commands

SENSe:DDEMod:MFILter:NAME

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:MFILter:NAME
value: str = driver.applications.k70Vsa.sense.ddemod.mfilter.name.get()
```

This command selects a measurement filter and automatically sets its state to 'ON'.

return
name: Name of the measurement filter or 'User' for a user-defined filter. An overview of available measurement filters is provided in 'Measurement filters'.

set(name: str) → None

```
# SCPI: [SENSe]:DDEMod:MFILter:NAME
driver.applications.k70Vsa.sense.ddemod.mfilter.name.set(name = '1')
```

This command selects a measurement filter and automatically sets its state to 'ON'.

param name
Name of the measurement filter or 'User' for a user-defined filter. An overview of available measurement filters is provided in 'Measurement filters'.

6.1.7.9.2.46 State

SCPI Commands

```
SENSe:DDEMod:MFILter:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:MFILter[:STATe]
value: bool = driver.applications.k70Vsa.sense.ddemod.mfilter.state.get()
```

Use this command to switch the measurement filter off. To switch a measurement filter on, use the [SENSe:]DDEMod:MFILter:NAME command.

return

state: OFF | 0 Switches the measurement filter off. ON | 1 Switches the measurement filter specified by [SENSe:]DDEMod:MFILter:NAME on. However, this command is not necessary, as the [SENSe:]DDEMod:MFILter:NAME command automatically switches the selected filter on.

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:MFILter[:STATe]
driver.applications.k70Vsa.sense.ddemod.mfilter.state.set(state = False)
```

Use this command to switch the measurement filter off. To switch a measurement filter on, use the [SENSe:]DDEMod:MFILter:NAME command.

param state

OFF | 0 Switches the measurement filter off. ON | 1 Switches the measurement filter specified by [SENSe:]DDEMod:MFILter:NAME on. However, this command is not necessary, as the [SENSe:]DDEMod:MFILter:NAME command automatically switches the selected filter on.

6.1.7.9.2.47 User

SCPI Commands

```
SENSe:DDEMod:MFILter:USER
```

class UserCls

User commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:MFILter:USER
value: str = driver.applications.k70Vsa.sense.ddemod.mfilter.user.get()
```

This command selects the user-defined measurement filter. For details on user-defined filters, see ‘Customized filters’.

return

filter_name: Name of the user-defined filter

set(filter_name: str) → None

```
# SCPI: [SENSe]:DDEMod:MFilter:USER
driver.applications.k70Vsa.sense.ddemod.mfilter.user.set(filter_name = '1')
```

This command selects the user-defined measurement filter. For details on user-defined filters, see ‘Customized filters’.

param filter_name

Name of the user-defined filter

6.1.7.9.2.48 Msk

class MskCls

Msk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.msk.clone()
```

Subgroups

6.1.7.9.2.49 FormatPy

SCPI Commands

```
SENSe:DDEMod:MSK:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → MskFormat

```
# SCPI: [SENSe]:DDEMod:MSK:FORMat
value: enums.MskFormat = driver.applications.k70Vsa.sense.ddemod.msk.formatPy.
↳get()
```

This command defines the specific demodulation order for MSK.

return

msk_format: TYPE1 | TYPE2 | NORMAl | DIFFerential TYPE1 | NORMAl Demodu-
lation order MSK is used. TYPE2 | DIFFerential Demodulation order DMSK is used.

set(msk_format: MskFormat) → None

```
# SCPI: [SENSe]:DDEMod:MSK:FORMat
driver.applications.k70Vsa.sense.ddemod.msk.formatPy.set(msk_format = enums.
↳MskFormat.DIFFerential)
```


This command defines the specific demodulation order for MSK.

param msk_format

TYPE1 | TYPE2 | NORMAl | DIFFerential TYPE1 | NORMAl Demodulation order MSK is used. TYPE2 | DIFFerential Demodulation order DMSK is used.

6.1.7.9.2.50 Normalize

class NormalizeCls

Normalize commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.normalize.clone()
```

Subgroups

6.1.7.9.2.51 Adroop

SCPI Commands

```
SENSe:DDEMod:NORMAlize:ADRoop
```

class AdroopCls

Adroop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMAlize:ADRoop
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.adroop.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMAlize:ADRoop
driver.applications.k70Vsa.sense.ddemod.normalize.adroop.set(state = False)
```

No command help available

param state

No help available

6.1.7.9.2.52 Cfdrift

SCPI Commands

`SENSe:DDEMod:NORMalize:CFDRift`

class CfdriftCls

Cfdrift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMalize:CFDRift
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.cfdrift.get()
```

This command defines whether the carrier frequency drift is compensated for FSK modulation.

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize:CFDRift
driver.applications.k70Vsa.sense.ddemod.normalize.cfdrift.set(state = False)
```

This command defines whether the carrier frequency drift is compensated for FSK modulation.

param state

No help available

6.1.7.9.2.53 Channel

SCPI Commands

`SENSe:DDEMod:NORMalize:CHANnel`

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMalize:CHANnel
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.channel.get()
```

This command switches the channel compensation on or off. (With equalizer only)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize:CHANnel
driver.applications.k70Vsa.sense.ddemod.normalize.channel.set(state = False)
```

This command switches the channel compensation on or off. (With equalizer only)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.54 FdError**SCPI Commands**

SENSe:DEMod:NORMalize:FDERror

class FdErrorCls

FdError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DEMod:NORMalize:FDERror
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.fdError.get()
```

This command defines whether the deviation error is compensated for when calculating the frequency error for FSK modulation.

return

state: ON | 1 Scales the reference signal to the actual deviation of the measurement signal. OFF | 0 Uses the entered nominal deviation for the reference signal.

set(state: bool) → None

```
# SCPI: [SENSe]:DEMod:NORMalize:FDERror
driver.applications.k70Vsa.sense.ddemod.normalize.fdError.set(state = False)
```

This command defines whether the deviation error is compensated for when calculating the frequency error for FSK modulation.

param state

ON | 1 Scales the reference signal to the actual deviation of the measurement signal.
OFF | 0 Uses the entered nominal deviation for the reference signal.

6.1.7.9.2.55 IqImbalance**SCPI Commands**

SENSe:DEMod:NORMalize:IQIMbalance

class IqImbalanceCls

IqImbalance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DEMod:NORMalize:IQIMbalance
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.iqImbalance.
    ↪get()
```

This command switches the compensation of the I/Q imbalance on or off.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize:IQIMbalance
driver.applications.k70Vsa.sense.ddemod.normalize.iqImbalance.set(state = False)
```

This command switches the compensation of the I/Q imbalance on or off.

param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.56 IqOffset

SCPI Commands

```
SENSe:DDEMod:NORMalize:IQOffset
```

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMalize:IQOffset
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.iqOffset.get()
```

This command switches the compensation of the I/Q offset on or off.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize:IQOffset
driver.applications.k70Vsa.sense.ddemod.normalize.iqOffset.set(state = False)
```

This command switches the compensation of the I/Q offset on or off.

param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.57 SrError

SCPI Commands

```
SENSe:DDEMod:NORMalize:SRERror
```

class SrErrorCls

SrError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMalize:SRERror
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.srError.get()
```

This command switches the compensation for symbol rate error on or off

```
return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function
on
```

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize:SRERror
driver.applications.k70Vsa.sense.ddemod.normalize.srError.set(state = False)
```

This command switches the compensation for symbol rate error on or off

```
param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on
```

6.1.7.9.2.58 Value

SCPI Commands

```
SENSe:DDEMod:NORMalize:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:NORMalize[:VALue]
value: bool = driver.applications.k70Vsa.sense.ddemod.normalize.value.get()
```

This command switches the compensation of the IQ offset and the compensation of amplitude droop on or off. Note that this command is maintained for compatibility reasons only. Use the more specific [SENSe:]DDEMod:NORMalize commands for new remote control programs (see ‘Demodulation settings’)

.

```
return
state: OFF | 0 No compensation for amplitude droop nor I/Q offset ON | 1 Compensa-
tion for amplitude droop and I/Q offset enabled
```

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:NORMalize[:VALue]
driver.applications.k70Vsa.sense.ddemod.normalize.value.set(state = False)
```

This command switches the compensation of the IQ offset and the compensation of amplitude droop on or off. Note that this command is maintained for compatibility reasons only. Use the more specific [SENSe:]DDEMod:NORMalize commands for new remote control programs (see ‘Demodulation settings’)

.

param state

OFF | 0 No compensation for amplitude droop nor I/Q offset ON | 1 Compensation for amplitude droop and I/Q offset enabled

6.1.7.9.2.59 Optimization

SCPI Commands

SENSe:DDEMod:OPTimization

class OptimizationCls

Optimization commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → OptimizationCriterion

<pre># SCPI: [SENSe]:DDEMod:OPTimization value: enums.OptimizationCriterion = driver.applications.k70Vsa.sense.ddemod. ↳ optimization.get()</pre>

This command determines the optimization criteria for the demodulation.

return

criterion: RMSMin | EVMMin RMSMin Optimizes calculation such that the RMS of the error vector is minimal. EVMMin Optimizes calculation such that EVM is minimal.

set(criterion: OptimizationCriterion) → None

<pre># SCPI: [SENSe]:DDEMod:OPTimization driver.applications.k70Vsa.sense.ddemod.optimization.set(criterion = enums. ↳ OptimizationCriterion.EVMMin)</pre>
--

This command determines the optimization criteria for the demodulation.

param criterion

RMSMin | EVMMin RMSMin Optimizes calculation such that the RMS of the error vector is minimal. EVMMin Optimizes calculation such that EVM is minimal.

6.1.7.9.2.60 Pattern

class PatternCls

Pattern commands group definition. 22 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.clone()
```

Subgroups

6.1.7.9.2.61 Apsk

class ApskCls

Apsk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.apsk.clone()
```

Subgroups

6.1.7.9.2.62 Nstate

SCPI Commands

```
SENSe:DDEMod:PATtern:APSK:NState
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:APSK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.apsk.nstate.get()
```

This command defines the demodulation order for APSK for the pattern (see also [SENSe:]DDEMod:PATtern:FORMat) . Depending on the demodulation state, the following orders are available:

Table Header: <APSKNstate> / Order

- 16 / 16APSK
- 32 / 32APSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return
apskn_state: 16 | 32

set(apskn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:APSK:NState
driver.applications.k70Vsa.sense.ddemod.pattern.apsk.nstate.set(apskn_state = 1.
↪0)
```

This command defines the demodulation order for APSK for the pattern (see also [SENSe:]DDEMod:PATtern:FORMat) . Depending on the demodulation state, the following orders are available:

Table Header: <APSKNstate> / Order

- 16 / 16APSK
- 32 / 32APSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param apskn_state
16 | 32

6.1.7.9.2.63 Ask

class AskCls

Ask commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.ask.clone()
```

Subgroups

6.1.7.9.2.64 Nstate

SCPI Commands

```
SENSe:DDEMod:PATtern:ASK:NState
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:ASK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.ask.nstate.get()
```

This command defines the demodulation order for ASK for the pattern (see also [SENSe:]DDEMod:PATtern:FORMat) . Depending on the demodulation state, the following orders are available:

Table Header: <ASKNstate> / Order

- 2 / 2ASK

- 4 / 4ASK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return
askn_state: 2 | 4

set(askn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:ASK:NState
driver.applications.k70Vsa.sense.ddemod.pattern.ask.nstate.set(askn_state = 1.0)
```

This command defines the demodulation order for ASK for the pattern (see also [SENSe:]DDEMod:PATtern:FORMat) . Depending on the demodulation state, the following orders are available:

Table Header: <ASKNstate> / Order

- 2 / 2ASK
- 4 / 4ASK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param askn_state
2 | 4

6.1.7.9.2.65 FormatPy

SCPI Commands

```
SENSe:DDEMod:PATtern:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DdemGroup

```
# SCPI: [SENSe]:DDEMod:PATtern:FORMat
value: enums.DdemGroup = driver.applications.k70Vsa.sense.ddemod.pattern.
↳ formatPy.get()
```

This command selects the pattern demodulation mode. Some modes can only be queried as they are not supported for the two modulations feature, but could be set when ‘Same as Data Symbols’ is selected.

return
group: MSK | PSK | QAM | QPSK | FSK | ASK | APSK | UQAM

set(group: DdemGroup) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FORMat
driver.applications.k70Vsa.sense.ddemod.pattern.formatPy.set(group = enums.
↳ DdemGroup.APSK)
```

This command selects the pattern demodulation mode. Some modes can only be queried as they are not supported for the two modulations feature, but could be set when ‘Same as Data Symbols’ is selected.

param group

MSK | PSK | QAM | QPSK | FSK | ASK | APSK | UQAM

6.1.7.9.2.66 Frame**class FrameCls**

Frame commands group definition. 10 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.frame.clone()
```

Subgroups**6.1.7.9.2.67 Edit****SCPI Commands**

```
SENSe:DDEMod:PATtern:FRAMe:EDIT
SENSe:DDEMod:PATtern:FRAMe:EDIT:SAVE
```

class EditCls

Edit commands group definition. 8 total commands, 4 Subgroups, 2 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.get()
```

Specifies an xml file for a user-defined frame structure configuration. The default storage location for such files is C:/R_S/INSTR/USER/vsa/FrameRangeStructure. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. If the specified file already exists, it is loaded for subsequent editing. Note that this command is a prerequisite to editing the frame structure of an existing file (using [SENSe:]DDEMod:PATtern:FRAMe:EDIT:STructure or any other command starting with [SENS:]DDEM:PATT:FRAM:EDIT) . It does not load the file for use in the current measurement (see [SENSe:]DDEMod:PATtern:FRAMe:LOAD) . Therefore, you can edit a frame structure while simultaneously performing a measurement with another frame structure configuration. If the file does not yet exist, a new frame structure is created and will be stored to the specified file when the [SENSe:]DDEMod:PATtern:FRAMe:EDIT:SAVE command is executed.

return

filename: string Path and file name of the xml file containing the frame structure configuration.

save(filename: Optional[str] = None) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:SAVE
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.save(filename = '1')
```

Stores the current frame structure configuration to the specified file. If no path is provided it is saved to the file selected previously by [SENSe:]DDEMod:PATtern:FRAMe:EDIT. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param filename

string Optional parameter: Path and file name of the xml file.

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.set(filename = '1')
```

Specifies an xml file for a user-defined frame structure configuration. The default storage location for such files is C:/R_S/INSTR/USER/vsa/FrameRangeStructure. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. If the specified file already exists, it is loaded for subsequent editing. Note that this command is a prerequisite to editing the frame structure of an existing file (using [SENSe:]DDEMod:PATtern:FRAMe:EDIT:STructure or any other command starting with [SENS:]DDEM:PATT:FRAM:EDIT) . It does not load the file for use in the current measurement (see [SENSe:]DDEMod:PATtern:FRAMe:LOAD) . Therefore, you can edit a frame structure while simultaneously performing a measurement with another frame structure configuration. If the file does not yet exist, a new frame structure is created and will be stored to the specified file when the [SENSe:]DDEMod:PATtern:FRAMe:EDIT:SAVE command is executed.

param filename

string Path and file name of the xml file containing the frame structure configuration.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.clone()
```

Subgroups

6.1.7.9.2.68 Next

class NextCls

Next commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.next.clone()
```

Subgroups

6.1.7.9.2.69 Boosting

SCPI Commands

SENSe:DDEMod:PATtern:FRAMe:EDIT:NEXT:BOOSting

class BoostingCls

Boosting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:NEXT:BOOSting
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.next.
↳boosting.get()
```

Determines which boosting is used to demodulate the frame next to the last configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

boosting: Range: 0.1 to 60

set(boosting: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:NEXT:BOOSting
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.next.boosting.
↳set(boosting = 1.0)
```

Determines which boosting is used to demodulate the frame next to the last configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param boosting

Range: 0.1 to 60

6.1.7.9.2.70 Modulation

SCPI Commands

SENSe:DDEMod:PATtern:FRAMe:EDIT:NEXT:MODulation

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FrameModulation

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:NEXT:MODulation
value: enums.FrameModulation = driver.applications.k70Vsa.sense.ddemod.pattern.
↳frame.edit.next.modulation.get()
```

Determines which modulation type is used to demodulate the frame after to the last configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

modulation: AUTO | DATA | PATtern Data The modulation type defined for data symbols is used (see [SENSe:]DDEMod:MAPPING[:VALue]) Pattern The modulation type defined for pattern symbols is used (see [SENSe:]DDEMod:PATtern:MAPPING[:VALue]) . Auto The nextt frame uses the same modulation as the first subframe of the frame configuration.

set(modulation: *FrameModulation*) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:NEXT:MODulation
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.next.modulation.
↪set(modulation = enums.FrameModulation.AUTO)
```

Determines which modulation type is used to demodulate the frame after to the last configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param modulation

AUTO | DATA | PATtern Data The modulation type defined for data symbols is used (see [SENSe:]DDEMod:MAPPING[:VALue]) Pattern The modulation type defined for pattern symbols is used (see [SENSe:]DDEMod:PATtern:MAPPING[:VALue]) . Auto The nextt frame uses the same modulation as the first subframe of the frame configuration.

6.1.7.9.2.71 Previous

class PreviousCls

Previous commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.previous.clone()
```

Subgroups

6.1.7.9.2.72 Boosting

SCPI Commands

```
SENSe:DDEMod:PATtern:FRAMe:EDIT:PREVIOUS:BOOSTing
```

class BoostingCls

Boosting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:FRaMe:EDIT:PREVious:BOOSting
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.
↳previous.boosting.get()
```

Determines which boosting is used to demodulate the frame previous to the first configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

boosting: Range: 0.1 to 60

set(boosting: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRaMe:EDIT:PREVious:BOOSting
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.previous.boosting.
↳set(boosting = 1.0)
```

Determines which boosting is used to demodulate the frame previous to the first configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param boosting

Range: 0.1 to 60

6.1.7.9.2.73 Modulation

SCPI Commands

```
SENSe:DDEMod:PATtern:FRaMe:EDIT:PREVious:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FrameModulation

```
# SCPI: [SENSe]:DDEMod:PATtern:FRaMe:EDIT:PREVious:MODulation
value: enums.FrameModulation = driver.applications.k70Vsa.sense.ddemod.pattern.
↳frame.edit.previous.modulation.get()
```

Determines which modulation type is used to demodulate the frame previous to the first configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

modulation: AUTO | DATA | PATtern Data The modulation type defined for data symbols is used (see [SENSe:]DDEMod:MAPPING[:VALue]) Pattern The modulation type defined for pattern symbols is used (see [SENSe:]DDEMod:PATtern:MAPPING[:VALue]) . Auto The previous frame uses the same modulation as the last subframe of the frame configuration.

set(modulation: FrameModulation) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:PREvious:MODulation
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.previous.modulation.
↪set(modulation = enums.FrameModulation.AUTO)
```

Determines which modulation type is used to demodulate the frame previous to the first configured subframe. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param modulation

AUTO | DATA | PATtern Data The modulation type defined for data symbols is used (see [SENSe:]DDEMod:MAPPing[:VALue]) Pattern The modulation type defined for pattern symbols is used (see [SENSe:]DDEMod:PATtern:MAPPing[:VALue]) . Auto The previous frame uses the same modulation as the last subframe of the frame configuration.

6.1.7.9.2.74 Structure

SCPI Commands

```
SENSe:DDEMod:PATtern:FRAMe:EDIT:STRUCTure
```

class StructureCls

Structure commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class StructureStruct

Structure for setting input parameters. Fields:

- Name: List[str]: string Name of the subframe. Duplicate names are allowed.
- Nof_Symbols: List[float]: integer The number of symbols the subframe consists of. For pattern subframes, the number of symbols must correspond to the number of symbols defined using [SENSe:]DDEMod:SEARch:SYNC:DATA.
- Modulation: List[enums.FrameModulationB]: DATA | PATtern Determines which modulation type is used to demodulate the subframe. The modulation for the 'previous frame' and 'next frame' are defined by separate commands (see [SENSe:]DDEMod:PATtern:FRAMe:EDIT:PREvious:MODulation and [SENSe:]DDEMod:PATtern:FRAMe:EDIT:NEXT:MODulation) . DATA The modulation type defined for data symbols is used (see [SENSe:]DDEMod:MAPPing[:VALue]) . PATtern The modulation type defined for patterns is used (see [SENSe:]DDEMod:PATtern:MAPPing[:VALue]) .
- Type_Py: List[enums.FrameModulationB]: DATA | PATtern Determines whether the demodulated data in the subframe is known or unknown by the R&S FSWP VSA application. PATtern The data is assumed to correspond with the pattern definition (see [SENSe:]DDEMod:SEARch:SYNC:DATA) . Not available for modulation type: 'DATA'. Only one subframe is allowed to be of type 'PATtern'. DATA The data is unknown. Used for data symbols or header information.
- Boosting: List[float]: numeric value For subframes with gain values different to the data symbols, define a different boosting factor to be applied to the reference power. Range: 0.1 to 60
- Description: List[str]: string Description for an individual subframe. Use an empty string ('') to leave out the description.

get() → StructureStruct

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:STRUCTure
value: StructureStruct = driver.applications.k70Vsa.sense.ddemod.pattern.frame.
↳edit.structure.get()
```

Defines the frame structure for a previously loaded file. For each subframe, all parameters must be defined. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. Note that the file must be loaded for editing before the structure can be defined using this command (see [SENSe:]DDEMod:PATtern:FRAMe:EDIT) . Loading the file for use in the current measurement is not sufficient (see [SENSe:]DDEMod:PATtern:FRAMe:LOAD) . Therefore, you can edit a frame structure while simultaneously performing a measurement with another frame structure configuration. The configuration is only stored by a subsequent [SENSe:]DDEMod:PATtern:FRAMe:EDIT:SAVE command. The modulation for the ‘previous frame’ and ‘next frame’ are defined by separate commands (see [SENS:]DDEM:PATT:FRAM:PREV:... and [SENS:]DDEM:PATT:FRAM:NEXT:...).

return

structure: for return value, see the help for StructureStruct structure arguments.

set(structure: StructureStruct) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:STRUCTure
structure = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.
↳structure.StructureStruct()
structure.Name: List[str] = ['1', '2', '3']
structure.Nof_Symbols: List[float] = [1.1, 2.2, 3.3]
structure.Modulation: List[enums.FrameModulationB] = [FrameModulationB.DATA,
↳FrameModulationB.PATtern]
structure.Type_Py: List[enums.FrameModulationB] = [FrameModulationB.DATA,
↳FrameModulationB.PATtern]
structure.Boosting: List[float] = [1.1, 2.2, 3.3]
structure.Description: List[str] = ['1', '2', '3']
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.structure.
↳set(structure)
```

Defines the frame structure for a previously loaded file. For each subframe, all parameters must be defined. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. Note that the file must be loaded for editing before the structure can be defined using this command (see [SENSe:]DDEMod:PATtern:FRAMe:EDIT) . Loading the file for use in the current measurement is not sufficient (see [SENSe:]DDEMod:PATtern:FRAMe:LOAD) . Therefore, you can edit a frame structure while simultaneously performing a measurement with another frame structure configuration. The configuration is only stored by a subsequent [SENSe:]DDEMod:PATtern:FRAMe:EDIT:SAVE command. The modulation for the ‘previous frame’ and ‘next frame’ are defined by separate commands (see [SENS:]DDEM:PATT:FRAM:PREV:... and [SENS:]DDEM:PATT:FRAM:NEXT:...).

param structure

for set value, see the help for StructureStruct structure arguments.

6.1.7.9.2.75 Text

SCPI Commands

SENSe:DDEMod:PATtern:FRAMe:EDIT:TEXT

class TextCls

Text commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:TEXT
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.text.
↳ get()
```

Defines the description for the frame structure in a previously loaded file. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. Note that the file must be loaded for editing before the description can be defined using this command (see [SENSe:]DDEMod:PATtern:FRAMe:EDIT).

return
filename: string

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:EDIT:TEXT
driver.applications.k70Vsa.sense.ddemod.pattern.frame.edit.text.set(filename =
↳ '1')
```

Defines the description for the frame structure in a previously loaded file. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. Note that the file must be loaded for editing before the description can be defined using this command (see [SENSe:]DDEMod:PATtern:FRAMe:EDIT).

param filename
string

6.1.7.9.2.76 Load

SCPI Commands

SENSe:DDEMod:PATtern:FRAMe:LOAD

class LoadCls

Load commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:LOAD
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.frame.load.get()
```

Loads a user-defined frame structure configuration to be used by the measurement from an xml file. The default storage location for such files is C:/R_S/INSTR/USER/vsa/FrameRangeStructure. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

filename: string Path and file name of the xml file. The default storage location for frame structures is C:/R_S/INSTR/USER/vsa/FrameRange_Structure.

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:LOAD
driver.applications.k70Vsa.sense.ddemod.pattern.frame.load.set(filename = '1')
```

Loads a user-defined frame structure configuration to be used by the measurement from an xml file. The default storage location for such files is C:/R_S/INSTR/USER/vsa/FrameRangeStructure. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param filename

string Path and file name of the xml file. The default storage location for frame structures is C:/R_S/INSTR/USER/vsa/FrameRange_Structure.

6.1.7.9.2.77 Mode**SCPI Commands**

```
SENSe:DDEMod:PATtern:FRAMe:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ConfigMode

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:MODE
value: enums.ConfigMode = driver.applications.k70Vsa.sense.ddemod.pattern.frame.
↳mode.get()
```

Determines whether the frame structure of the signal is configured in reference to the result range or user-defined. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

frame_mode: DEFAULT | USER Default A single frame is assumed to correspond to the result range defined by [SENSe:]DDEMod:TIME. User-Defined A frame is defined manually as a succession of subframes with specified characteristics. In this case, the result range is assumed to be a single frame as specified by [SENSe:]DDEMod:PATtern:FRAMe:EDIT:STRucture. If no structure is configured or loaded yet, the result range definition is used (as for 'Default').

set(frame_mode: ConfigMode) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:FRAMe:MODE
driver.applications.k70Vsa.sense.ddemod.pattern.frame.mode.set(frame_mode =
↳enums.ConfigMode.DEFAULT)
```

Determines whether the frame structure of the signal is configured in reference to the result range or user-defined. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param frame_mode

Default | USER Default A single frame is assumed to correspond to the result range defined by [SENSe:]DDEMod:TIME. User-Defined A frame is defined manually as a succession of subframes with specified characteristics. In this case, the result range is assumed to be a single frame as specified by [SENSe:]DDEMod:PATtern:FRAME:EDIT:STRucture. If no structure is configured or loaded yet, the result range definition is used (as for 'Default').

6.1.7.9.2.78 Mapping**class MappingCls**

Mapping commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.mapping.clone()
```

Subgroups**6.1.7.9.2.79 Catalog****SCPI Commands**

```
SENSe:DDEMod:PATtern:MAPPing:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:MAPPing:CATalog
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.mapping.catalog.
    ↪ get()
```

This command queries the names of all mappings that are available for the pattern for the current modulation type and order. A mapping describes the assignment of constellation points to symbols. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

mappings: list A comma-separated list of strings, with one string for each mapping name.

6.1.7.9.2.80 Value

SCPI Commands

`SENSe:DDEMod:PATtern:MAPPing:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:MAPPing[:VALue]
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.mapping.value.get()
```

This command selects the mapping for pattern demodulation. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

mapping: To obtain a list of available symbol mappings for the current modulation type use the [SENSe:]DDEMod:PATtern:MAPPing:CATalog?? query.

set(mapping: str) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:MAPPing[:VALue]
driver.applications.k70Vsa.sense.ddemod.pattern.mapping.value.set(mapping = '1')
```

This command selects the mapping for pattern demodulation. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param mapping

To obtain a list of available symbol mappings for the current modulation type use the [SENSe:]DDEMod:PATtern:MAPPing:CATalog?? query.

6.1.7.9.2.81 Psk

class PskCls

Psk commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.psk.clone()
```

Subgroups

6.1.7.9.2.82 FormatPy

SCPI Commands

```
SENSe:DDEMod:PATtern:PSK:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PskFormat

```
# SCPI: [SENSe]:DDEMod:PATtern:PSK:FORMat
value: enums.PskFormat = driver.applications.k70Vsa.sense.ddemod.pattern.psk.
    ↪ formatPy.get()
```

Together with DDEMod:PATT:PSK:NST, this command defines the demodulation order for PSK for the pattern (see also [SENSe:]DDEMod:PATtern:PSK:NSTate).

Table Header: NSTate / <PSKformat> / Order

- 2 / NORMal / BPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

psk_format: NORMal | DIFFerential

set(psk_format: PskFormat) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:PSK:FORMat
driver.applications.k70Vsa.sense.ddemod.pattern.psk.formatPy.set(psk_format =
    ↪ enums.PskFormat.DIFFerential)
```

Together with DDEMod:PATT:PSK:NST, this command defines the demodulation order for PSK for the pattern (see also [SENSe:]DDEMod:PATtern:PSK:NSTate).

Table Header: NSTate / <PSKformat> / Order

- 2 / NORMal / BPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param psk_format

NORMal | DIFFerential

6.1.7.9.2.83 Nstate

SCPI Commands

SENSe:DDEMod:PATtern:PSK:NState

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:PSK:NState
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.psk.nstate.get()
```

Together with DDEMod:PATT:PSK:FORMat, this command defines the demodulation order for PSK for the pattern (see also [SENSe:]DDEMod:PATtern:PSK:FORMat) . Depending on the demodulation format and state, the following orders are available:

Table Header: <PSKNSTATE> / FORMat / Order

- 2 / any / BPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

```
return
pskn_state: 2 | 8
```

set(pskn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:PSK:NState
driver.applications.k70Vsa.sense.ddemod.pattern.psk.nstate.set(pskn_state = 1.0)
```

Together with DDEMod:PATT:PSK:FORMat, this command defines the demodulation order for PSK for the pattern (see also [SENSe:]DDEMod:PATtern:PSK:FORMat) . Depending on the demodulation format and state, the following orders are available:

Table Header: <PSKNSTATE> / FORMat / Order

- 2 / any / BPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

```
param pskn_state
2 | 8
```

6.1.7.9.2.84 Qam

class QamCls

Qam commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.qam.clone()
```

Subgroups

6.1.7.9.2.85 FormatPy

SCPI Commands

```
SENSe:DDEMod:PATtern:QAM:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → QamFormat

```
# SCPI: [SENSe]:DDEMod:PATtern:QAM:FORMat
value: enums.QamFormat = driver.applications.k70Vsa.sense.ddemod.pattern.qam.
↳ formatPy.get()
```

This command defines the specific demodulation order for QAM for the pattern. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

qam_format: NORMal | DIFFerential NORMal Demodulation order QAM is used.
DIFFerential Demodulation order DQAM is used.

set(qam_format: QamFormat) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:QAM:FORMat
driver.applications.k70Vsa.sense.ddemod.pattern.qam.formatPy.set(qam_format =
↳ enums.QamFormat.DIFFerential)
```

This command defines the specific demodulation order for QAM for the pattern. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param qam_format

NORMal | DIFFerential NORMal Demodulation order QAM is used. DIFFerential Demodulation order DQAM is used.

6.1.7.9.2.86 Nstate

SCPI Commands

`SENSe:DDEMod:PATtern:QAM:NState`

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PATtern:QAM:NState
value: float = driver.applications.k70Vsa.sense.ddemod.pattern.qam.nstate.get()
```

This command defines the demodulation order for QAM for the pattern.

Table Header: <QAMNState> / Order

- 16 / 16QAM
- 16 / Pi/4-16QAM
- 32 / 32QAM
- 32 / Pi/4-32QAM
- 64 / 64QAM
- 128 / 128QAM
- 256 / 256QAM
- 512 / 512QAM
- 1024 / 1024QAM

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

qamn_state: No help available

set(qamn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:QAM:NState
driver.applications.k70Vsa.sense.ddemod.pattern.qam.nstate.set(qamn_state = 1.0)
```

This command defines the demodulation order for QAM for the pattern.

Table Header: <QAMNState> / Order

- 16 / 16QAM
- 16 / Pi/4-16QAM
- 32 / 32QAM
- 32 / Pi/4-32QAM
- 64 / 64QAM
- 128 / 128QAM

- 256 / 256QAM
- 512 / 512QAM
- 1024 / 1024QAM

This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param qamn_state
No help available

6.1.7.9.2.87 Qpsk

class QpskCls

Qpsk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.qpsk.clone()
```

Subgroups

6.1.7.9.2.88 FormatPy

SCPI Commands

```
SENSe:DDEMod:PATtern:QPSK:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → QpskFormat

```
# SCPI: [SENSe]:DDEMod:PATtern:QPSK:FORMat
value: enums.QpskFormat = driver.applications.k70Vsa.sense.ddemod.pattern.qpsk.
↳formatPy.get()
```

This command defines the demodulation order for QPSK for the pattern. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return
qpsk_format: NORMal | DIFFerential NORMal Demodulation order QPSK is used.
DIFFerential Demodulation order DQPSK is used.

set(qpsk_format: QpskFormat) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:QPSK:FORMat
driver.applications.k70Vsa.sense.ddemod.pattern.qpsk.formatPy.set(qpsk_format =
↳enums.QpskFormat.DIFFerential)
```

This command defines the demodulation order for QPSK for the pattern. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param qpsk_format

NORMAL | DIFFerential NORMAL Demodulation order QPSK is used. DIFFerential Demodulation order DQPSK is used.

6.1.7.9.2.89 State

SCPI Commands

SENSe:DDEMod:PATtern:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

<pre># SCPI: [SENSe]:DDEMod:PATtern[:STATe] value: bool = driver.applications.k70Vsa.sense.ddemod.pattern.state.get()</pre>

Determines whether the pattern uses a different modulation type than the data symbols. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

state: ON | OFF | 0 | 1 OFF | 0 The pattern uses the same modulation as the data symbols, defined by [SENSe:]DDEMod:MAPPing[:VALue]. ON | 1 The pattern uses a different modulation, configured by [SENSe:]DDEMod:PATtern:MAPPing[:VALue].

set(state: bool) → None

<pre># SCPI: [SENSe]:DDEMod:PATtern[:STATe] driver.applications.k70Vsa.sense.ddemod.pattern.state.set(state = False)</pre>
--

Determines whether the pattern uses a different modulation type than the data symbols. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param state

ON | OFF | 0 | 1 OFF | 0 The pattern uses the same modulation as the data symbols, defined by [SENSe:]DDEMod:MAPPing[:VALue]. ON | 1 The pattern uses a different modulation, configured by [SENSe:]DDEMod:PATtern:MAPPing[:VALue].

6.1.7.9.2.90 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.pattern.user.clone()
```

Subgroups

6.1.7.9.2.91 Name

SCPI Commands

```
SENSe:DDEMod:PATtern:USER:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:PATtern:USER:NAME
value: str = driver.applications.k70Vsa.sense.ddemod.pattern.user.name.get()
```

Selects the file that contains a user-defined modulation. For details on user-defined modulation files see ‘User-defined modulation’. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. The default storage location for user-defined modulations is C:/R_S/INSTR/USER/vsa/Constellation.

return

name: string Path and file name of the *.vam file.

set(name: str) → None

```
# SCPI: [SENSe]:DDEMod:PATtern:USER:NAME
driver.applications.k70Vsa.sense.ddemod.pattern.user.name.set(name = '1')
```

Selects the file that contains a user-defined modulation. For details on user-defined modulation files see ‘User-defined modulation’. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed. The default storage location for user-defined modulations is C:/R_S/INSTR/USER/vsa/Constellation.

param name

string Path and file name of the *.vam file.

6.1.7.9.2.92 Prate

SCPI Commands

```
SENSe:DDEMod:PRATe
```

class PrateCls

Prate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PRATe
value: float = driver.applications.k70Vsa.sense.ddemod.prate.get()
```

Defines the number of samples that are captured per symbol, i.e. the factor by which the symbol rate is multiplied to obtain the sample rate. This parameter also affects the demodulation bandwidth and thus the usable I/Q bandwidth. The sample rate depends on the defined ‘Symbol Rate’ (see ‘Sample rate, symbol rate and I/Q bandwidth’).

return

capt_oversampling: | 2 | 4 | 8 | 16 | 32 | 64 | 128 The factor by which the symbol rate is multiplied to obtain the sample rate, e.g. 4 samples per symbol: sample rate = 4*symbol rate

set(capt_oversampling: float) → None

```
# SCPI: [SENSe]:DDEMod:PRATe
driver.applications.k70Vsa.sense.ddemod.prate.set(capt_oversampling = 1.0)
```

Defines the number of samples that are captured per symbol, i.e. the factor by which the symbol rate is multiplied to obtain the sample rate. This parameter also affects the demodulation bandwidth and thus the usable I/Q bandwidth. The sample rate depends on the defined ‘Symbol Rate’ (see ‘Sample rate, symbol rate and I/Q bandwidth’).

param capt_oversampling

2 | 4 | 8 | 16 | 32 | 64 | 128 The factor by which the symbol rate is multiplied to obtain the sample rate, e.g. 4 samples per symbol: sample rate = 4*symbol rate

6.1.7.9.2.93 Preset

class PresetCls

Preset commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.preset.clone()
```

Subgroups

6.1.7.9.2.94 Calc

SCPI Commands

```
SENSe:DDEMod:PRESet:CALC
```

class CalcCls

Calc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:DDEMod:PRESet:CALC
driver.applications.k70Vsa.sense.ddemod.preset.calc.set()
```

This command selects a predefined ‘signal overview’ consisting of four windows. The top left window (1) shows magnitude data from capture buffer, the top right window (2) spectrum data from capture buffer, the bottom left window (3) the ‘Result Summary’ and the bottom right window (4) constellation I/Q data. Using this setup, scripts written for R&S FSV instruments will continue to work.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:DDEMod:PRESet:CALC
driver.applications.k70Vsa.sense.ddemod.preset.calc.set_with_opc()
```

This command selects a predefined ‘signal overview’ consisting of four windows. The top left window (1) shows magnitude data from capture buffer, the top right window (2) spectrum data from capture buffer, the bottom left window (3) the ‘Result Summary’ and the bottom right window (4) constellation I/Q data. Using this setup, scripts written for R&S FSV instruments will continue to work.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.9.2.95 RefLevel

SCPI Commands

```
SENSe:DDEMod:PRESet:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:DDEMod:PRESet:RLEVel
driver.applications.k70Vsa.sense.ddemod.preset.refLevel.set()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:DDEMod:PRESet:RLEVel
driver.applications.k70Vsa.sense.ddemod.preset.refLevel.set_with_opc()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.9.2.96 Standard

SCPI Commands

SENSe:DDEMod:PRESet:STANdard

class StandardCls

Standard commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TechnologyStandardDdem

```
# SCPI: [SENSe]:DDEMod:PRESet[:STANdard]
value: enums.TechnologyStandardDdem = driver.applications.k70Vsa.sense.ddemod.
↳ preset.standard.get()
```

This command selects an automatic setting of all modulation parameters according to a standardized transmission method or a user-defined transmission method. The standardized transmission methods are available in the instrument as predefined standards.

return

standard: (enum or string) Specifies the file name that contains the transmission method without the extension. For user-defined standards, the file path must be included. Default standards predefined by Rohde&Schwarz do not require a path definition. A list of predefined standards (including short forms) is provided in the annex (see ‘Predefined standards and settings’).

set(standard: TechnologyStandardDdem) → None

```
# SCPI: [SENSe]:DDEMod:PRESet[:STANdard]
driver.applications.k70Vsa.sense.ddemod.preset.standard.set(standard = enums.
↳ TechnologyStandardDdem.DECT)
```

This command selects an automatic setting of all modulation parameters according to a standardized transmission method or a user-defined transmission method. The standardized transmission methods are available in the instrument as predefined standards.

param standard

(enum or string) Specifies the file name that contains the transmission method without the extension. For user-defined standards, the file path must be included. Default standards predefined by Rohde&Schwarz do not require a path definition. A list of predefined standards (including short forms) is provided in the annex (see ‘Predefined standards and settings’).

6.1.7.9.2.97 Psk

class PskCls

Psk commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.psk.clone()
```

Subgroups

6.1.7.9.2.98 FormatPy

SCPI Commands

```
SENSe:DDEMod:PSK:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PskFormat

```
# SCPI: [SENSe]:DDEMod:PSK:FORMat
value: enums.PskFormat = driver.applications.k70Vsa.sense.ddemod.psk.formatPy.
↳get()
```

Together with DDEMod:PSK:NST, this command defines the demodulation order for PSK (see also [SENSe:]DDEMod:PSK:NSTate) . Depending on the demodulation format and state, the following orders are available:

Table Header: NSTate / <PSKformat> / Order

- 2 / NORMal / BPSK
- 2 / NPI2 / Pi/2-BPSK
- 2 / MNPI2 / Pi/2-BPSK
- 2 / DPI2 / Pi/2-DBPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK
- 8 / N3Pi8 / 3pi/8-8PSK (EDGE)
- 8 / PI8D8PSK / Pi/8-D8PSK

return

psk_format: NORMal | DIFFerential | N3Pi8 | PI8D8psk | NPI2 | DPI2 | MNPI2

set(psk_format: PskFormat) → None

```
# SCPI: [SENSe]:DDEMod:PSK:FORMat
driver.applications.k70Vsa.sense.ddemod.psk.formatPy.set(psk_format = enums.
↳ PskFormat.DIFFerential)
```

Together with DDEMod:PSK:NST, this command defines the demodulation order for PSK (see also [SENSe:]DDEMod:PSK:NSTate) . Depending on the demodulation format and state, the following orders are available:

Table Header: NSTate / <PSKformat> / Order

- 2 / NORMal / BPSK
- 2 / NPI2 / Pi/2-BPSK
- 2 / MNPI2 / Pi/2-BPSK
- 2 / DPI2 / Pi/2-DBPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK
- 8 / N3Pi8 / 3pi/8-8PSK (EDGE)
- 8 / PI8D8PSK / Pi/8-D8PSK

param psk_format

NORMal | DIFFerential | N3Pi8 | PI8D8psk | NPI2 | DPI2 | MNPI2

6.1.7.9.2.99 Nstate

SCPI Commands

```
SENSe:DDEMod:PSK:NSTate
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:PSK:NSTate
value: float = driver.applications.k70Vsa.sense.ddemod.psk.nstate.get()
```

Together with DDEMod:PSK:FORMat, this command defines the demodulation order for PSK (see also [SENSe:]DDEMod:PSK:FORMat) . Depending on the demodulation format and state, the following orders are available:

Table Header: <PSKNSTate> / FORMat / Order

- 2 / any / BPSK
- 2 / NPI2 / Pi/2-BPSK
- 2 / DPI2 / Pi/2-DBPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK
- 8 / N3Pi8 / 3pi/8-8PSK (EDGE)

- 8 / PI8D8PSK / Pi/8-D8PSK

return
pskn_state: 2 | 8

set(pskn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:PSK:NState
driver.applications.k70Vsa.sense.ddemod.psk.nstate.set(pskn_state = 1.0)
```

Together with DDEMod:PSK:FORMat, this command defines the demodulation order for PSK (see also [SENSe]:DDEMod:PSK:FORMat). Depending on the demodulation format and state, the following orders are available:

Table Header: <PSKNSTATe> / FORMat / Order

- 2 / any / BPSK
- 2 / NPI2 / Pi/2-BPSK
- 2 / DPI2 / Pi/2-DBPSK
- 8 / NORMal / 8PSK
- 8 / DIFFerential / D8PSK
- 8 / N3Pi8 / 3pi/8-8PSK (EDGE)
- 8 / PI8D8PSK / Pi/8-D8PSK

param pskn_state
2 | 8

6.1.7.9.2.100 Qam

class QamCls

Qam commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.qam.clone()
```

Subgroups

6.1.7.9.2.101 FormatPy

SCPI Commands

```
SENSe:DDEMod:QAM:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → QamFormat

```
# SCPI: [SENSe]:DDEMod:QAM:FORMat
value: enums.QamFormat = driver.applications.k70Vsa.sense.ddemod.qam.formatPy.
↳ get()
```

This command defines the specific demodulation order for QAM.

return

qam_format: NORMal | DIFFerential | NPI4 | MNPI4 NORMal Demodulation order QAM is used. DIFFerential Demodulation order DQAM is used. NPI4 Demodulation order /4-16QAM is used. MNPI4 Demodulation order -/4-32QAM is used.

set(qam_format: QamFormat) → None

```
# SCPI: [SENSe]:DDEMod:QAM:FORMat
driver.applications.k70Vsa.sense.ddemod.qam.formatPy.set(qam_format = enums.
↳ QamFormat.DIFFerential)
```

This command defines the specific demodulation order for QAM.

param qam_format

NORMal | DIFFerential | NPI4 | MNPI4 NORMal Demodulation order QAM is used. DIFFerential Demodulation order DQAM is used. NPI4 Demodulation order /4-16QAM is used. MNPI4 Demodulation order -/4-32QAM is used.

6.1.7.9.2.102 Nstate

SCPI Commands

SENSe:DDEMod:QAM:NState

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:QAM:NState
value: float = driver.applications.k70Vsa.sense.ddemod.qam.nstate.get()
```

This command defines the demodulation order for QAM.

Table Header: <QAMNState> / Order

- 16 / 16QAM
- 16 / Pi/4-16QAM
- 32 / 32QAM
- 32 / Pi/4-32QAM
- 64 / 64QAM
- 128 / 128QAM
- 256 / 256QAM
- 512 / 512QAM

- 1024 / 1024QAM

return

qamn_state: No help available

set(qamn_state: float) → None

```
# SCPI: [SENSe]:DDEMod:QAM:NState
driver.applications.k70Vsa.sense.ddemod.qam.nstate.set(qamn_state = 1.0)
```

This command defines the demodulation order for QAM.

Table Header: <QAMNState> / Order

- 16 / 16QAM
- 16 / Pi/4-16QAM
- 32 / 32QAM
- 32 / Pi/4-32QAM
- 64 / 64QAM
- 128 / 128QAM
- 256 / 256QAM
- 512 / 512QAM
- 1024 / 1024QAM

param qamn_state

No help available

6.1.7.9.2.103 Qpsk

class QpskCls

Qpsk commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.qpsk.clone()
```

Subgroups

6.1.7.9.2.104 FormatPy

SCPI Commands

```
SENSe:DDEMod:QPSK:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → QpskFormat

```
# SCPI: [SENSe]:DDEMod:QPSK:FORMat
value: enums.QpskFormat = driver.applications.k70Vsa.sense.ddemod.qpsk.formatPy.
↪get()
```

This command defines the demodulation order for QPSK.

return

qpsk_format: NORMal | DIFFerential | NPI4 | DPI4 | OFFSet | SOFFset | N3Pi4 NOR-
Mal Demodulation order QPSK is used. DIFFerential Demodulation order DQPSK
is used. NPI4 Demodulation order /4 QPSK is used. DPI4 Demodulation order /4
DQPSK is used. OFFSet Demodulation order OQPSK is used. N3PI4 Demodulation
order 3/4 QPSK is used. SOFFset Shaped Offset QPSK

set(qpsk_format: QpskFormat) → None

```
# SCPI: [SENSe]:DDEMod:QPSK:FORMat
driver.applications.k70Vsa.sense.ddemod.qpsk.formatPy.set(qpsk_format = enums.
↪QpskFormat.DIFFerential)
```

This command defines the demodulation order for QPSK.

param qpsk_format

NORMal | DIFFerential | NPI4 | DPI4 | OFFSet | SOFFset | N3Pi4 NORMal Demod-
ulation order QPSK is used. DIFFerential Demodulation order DQPSK is used. NPI4
Demodulation order /4 QPSK is used. DPI4 Demodulation order /4 DQPSK is used.
OFFSet Demodulation order OQPSK is used. N3PI4 Demodulation order 3/4 QPSK
is used. SOFFset Shaped Offset QPSK

6.1.7.9.2.105 Rlength

class RlengthCls

Rlength commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.rlength.clone()
```

Subgroups

6.1.7.9.2.106 Auto

SCPI Commands

```
SENSe:DDEMod:RELEnGth:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:RENgth:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.rlength.auto.get()
```

If enabled, the capture length is automatically adapted as required according to the current result length, burst and pattern search settings, and network-specific characteristics (e.g. burst and frame structures) .

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:RENgth:AUTO
driver.applications.k70Vsa.sense.ddemod.rlength.auto.set(state = False)
```

If enabled, the capture length is automatically adapted as required according to the current result length, burst and pattern search settings, and network-specific characteristics (e.g. burst and frame structures) .

param state

No help available

6.1.7.9.2.107 Symbols

class SymbolsCls

Symbols commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.rlength.symbols.clone()
```

Subgroups

6.1.7.9.2.108 Value

SCPI Commands

```
SENSe:DDEMod:RENgth:SYMBols:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:RENgth:SYMBols[:VALue]
value: float = driver.applications.k70Vsa.sense.ddemod.rlength.symbols.value.
    ↪get()
```

This command defines the capture length for further processing, e.g. for burst search, in symbols. Note that the maximum record length depends on the sample rate for signal capture (see [SENSe:]DDEMod:PRATe). The maximum record length (in symbols) can be calculated as: $\text{RecordlengthMAX} = 460000000 / \text{<points per symbol>}$

```
return
    record_length: Unit: SYM
```

```
set(record_length: float) → None
```

```
# SCPI: [SENSe]:DDEMod:RENgth:SYMBOLs[:VALue]
driver.applications.k70Vsa.sense.ddemod.rlength.symbols.value.set(record_length,
↪= 1.0)
```

This command defines the capture length for further processing, e.g. for burst search, in symbols. Note that the maximum record length depends on the sample rate for signal capture (see [SENSe:]DDEMod:PRATe). The maximum record length (in symbols) can be calculated as: $\text{RecordlengthMAX} = 460000000 / \text{<points per symbol>}$

```
param record_length
    Unit: SYM
```

6.1.7.9.2.109 Value

SCPI Commands

```
SENSe:DDEMod:RENgth:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get() → float
```

```
# SCPI: [SENSe]:DDEMod:RENgth[:VALue]
value: float = driver.applications.k70Vsa.sense.ddemod.rlength.value.get()
```

This command defines or queries the capture length for further processing, e.g. for burst search. Note that the maximum capture length depends on the sample rate for signal capture (see [SENSe:]DDEMod:PRATe).

```
return
    record_length: The capture length can be defined in time (seconds) or symbols (SYM)
    . The return value is always in time (s) . To query the capture length in symbols, use
    the [SENSe:]DDEMod:RENgth:SYMBOLs[:VALue] command. Unit: S
```

```
set(record_length: float) → None
```

```
# SCPI: [SENSe]:DDEMod:RENgth[:VALue]
driver.applications.k70Vsa.sense.ddemod.rlength.value.set(record_length = 1.0)
```

This command defines or queries the capture length for further processing, e.g. for burst search. Note that the maximum capture length depends on the sample rate for signal capture (see [SENSe:]DDEMod:PRATe).

param record_length

The capture length can be defined in time (seconds) or symbols (SYM) . The return value is always in time (s) . To query the capture length in symbols, use the [SENSe:]DDEMod:RLEnGth:SYMBols[:VALue] command. Unit: S

6.1.7.9.2.110 Sband**SCPI Commands**

SENSe:DDEMod:SBANd

class SbandCls

Sband commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SidebandPos

<pre># SCPI: [SENSe]:DDEMod:SBANd value: enums.SidebandPos = driver.applications.k70Vsa.sense.ddemod.sband.get()</pre>
--

This command selects the sideband for the demodulation. Note that this command is maintained for compatibility reasons only. Use the SENS:SWAP:IQ command for new remote control programs (see [SENSe:]SWAPiq) .

return

sideband_pos: NORMal | INVerse NORMal Normal (non-inverted) position INVerse
Inverted position

set(sideband_pos: SidebandPos) → None

<pre># SCPI: [SENSe]:DDEMod:SBANd driver.applications.k70Vsa.sense.ddemod.sband.set(sideband_pos = enums. ↪SidebandPos.INVerse)</pre>

This command selects the sideband for the demodulation. Note that this command is maintained for compatibility reasons only. Use the SENS:SWAP:IQ command for new remote control programs (see [SENSe:]SWAPiq) .

param sideband_pos

NORMal | INVerse NORMal Normal (non-inverted) position INVerse Inverted position

6.1.7.9.2.111 Search**class SearchCls**

Search commands group definition. 36 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.clone()
```

Subgroups

6.1.7.9.2.112 Burst

class BurstCls

Burst commands group definition. 10 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.burst.clone()
```

Subgroups

6.1.7.9.2.113 Auto

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:AUTO
value: enums.AutoManualMode = driver.applications.k70Vsa.sense.ddemod.search.
↳ burst.auto.get()
```

This command links the burst search to the type of signal. When a signal is marked as bursted, burst search is switched on automatically.

return

auto_burst_search: AUTO | MANual

set(auto_burst_search: AutoManualMode) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:AUTO
driver.applications.k70Vsa.sense.ddemod.search.burst.auto.set(auto_burst_search,
↳ enums.AutoManualMode.AUTO)
```

This command links the burst search to the type of signal. When a signal is marked as bursted, burst search is switched on automatically.

param auto_burst_search

AUTO | MANual

6.1.7.9.2.114 Configure

class ConfigureCls

Configure commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.burst.configure.clone()
```

Subgroups

6.1.7.9.2.115 Auto

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:CONFigure:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:CONFigure:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.search.burst.configure.
↳ auto.get()
```

This command sets the search tolerance and the min gap length to their default values.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:CONFigure:AUTO
driver.applications.k70Vsa.sense.ddemod.search.burst.configure.auto.set(state =
↳ False)
```

This command sets the search tolerance and the min gap length to their default values.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.116 Glength

class GlengthCls

Glength commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.burst.glength.clone()
```

Subgroups

6.1.7.9.2.117 Minimum

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:GLENgth:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:GLENgth[:MINimum]
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.glength.
↳ minimum.get()
```

This command defines the minimum time between two bursts. A minimum time with decreased level must occur between two bursts. The default unit is symbol. The value can also be given in seconds.

return

min_gap_length: Range: 1 to 15000, Unit: SYM

set(min_gap_length: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:GLENgth[:MINimum]
driver.applications.k70Vsa.sense.ddemod.search.burst.glength.minimum.set(min_
↳ gap_length = 1.0)
```

This command defines the minimum time between two bursts. A minimum time with decreased level must occur between two bursts. The default unit is symbol. The value can also be given in seconds.

param min_gap_length

Range: 1 to 15000, Unit: SYM

6.1.7.9.2.118 Length

class LengthCls

Length commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.burst.length.clone()
```

Subgroups

6.1.7.9.2.119 Maximum

SCPI Commands

```
SENSe:DDEMod:SEARCh:BURSt:LENGth:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARCh:BURSt:LENGth:MAXimum
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.length.
↳maximum.get()
```

No command help available

return
max_length: Range: 0 to 128000, Unit: SYM

set(max_length: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:BURSt:LENGth:MAXimum
driver.applications.k70Vsa.sense.ddemod.search.burst.length.maximum.set(max_
↳length = 1.0)
```

No command help available

param max_length
Range: 0 to 128000, Unit: SYM

6.1.7.9.2.120 Minimum

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:LENGth:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:LENGth[:MINimum]
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.length.
↳ minimum.get()
```

This command defines the minimum useful length of a burst. Only those bursts will be recognized that exceed this length. The default unit is symbols. The value can also be given in seconds. Note the difference to manual operation: <Min_length>Manual= <Min_Useful_Length> + <Run-In> + <Run-Out>

return

useful_length: numeric value Range: 10 to 32000, Unit: Sym

set(useful_length: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:LENGth[:MINimum]
driver.applications.k70Vsa.sense.ddemod.search.burst.length.minimum.set(useful_
↳ length = 1.0)
```

This command defines the minimum useful length of a burst. Only those bursts will be recognized that exceed this length. The default unit is symbols. The value can also be given in seconds. Note the difference to manual operation: <Min_length>Manual= <Min_Useful_Length> + <Run-In> + <Run-Out>

param useful_length

numeric value Range: 10 to 32000, Unit: Sym

6.1.7.9.2.121 Mode

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → BurstMode

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:MODE
value: enums.BurstMode = driver.applications.k70Vsa.sense.ddemod.search.burst.
↳ mode.get()
```

This command sets the vector analyzer so that a measurement is performed only if a burst is found. The command is available only if the burst search is activated (see [SENSe:]DDEMod:SEARch:BURSt:STATe).

return

meas_only_on_burst: MEAS | BURS MEAS Measurement is always performed
BURS Measurement is performed only if a burst is found

set(meas_only_on_burst: BurstMode) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:MODE
driver.applications.k70Vsa.sense.ddemod.search.burst.mode.set(meas_only_on_
↳burst = enums.BurstMode.BURS)
```

This command sets the vector analyzer so that a measurement is performed only if a burst is found. The command is available only if the burst search is activated (see [SENSe:]DDEMod:SEARch:BURSt:STATe).

param meas_only_on_burst

MEAS | BURS MEAS Measurement is always performed BURS Measurement is performed only if a burst is found

6.1.7.9.2.122 Skip

class SkipCls

Skip commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.burst.skip.clone()
```

Subgroups

6.1.7.9.2.123 Falling

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:SKIP:FALLing
```

class FallingCls

Falling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:SKIP:FALLing
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.skip.
↳falling.get()
```

This command defines the length of the falling burst edge which is not considered when evaluating the result.

return

run_out: Range: 1, Unit: SYM

set(*run_out: float*) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:SKIP:FALLing
driver.applications.k70Vsa.sense.ddemod.search.burst.skip.falling.set(run_out = 1.0)
```

This command defines the length of the falling burst edge which is not considered when evaluating the result.

param run_out

Range: 1 , Unit: SYM

6.1.7.9.2.124 Rising

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:SKIP:RISing
```

class RisingCls

Rising commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:SKIP:RISing
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.skip.rising.get()
```

No command help available

return

run_in: Range: 0 to 31990, Unit: SYM

set(*run_in: float*) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:SKIP:RISing
driver.applications.k70Vsa.sense.ddemod.search.burst.skip.rising.set(run_in = 1.0)
```

No command help available

param run_in

Range: 0 to 31990, Unit: SYM

6.1.7.9.2.125 State

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:STAtE
value: bool = driver.applications.k70Vsa.sense.ddemod.search.burst.state.get()
```

This command switches the search for a signal burst on or off.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:STAtE
driver.applications.k70Vsa.sense.ddemod.search.burst.state.set(state = False)
```

This command switches the search for a signal burst on or off.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.126 Tolerance

SCPI Commands

```
SENSe:DDEMod:SEARch:BURSt:TOLerance
```

class ToleranceCls

Tolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:TOLerance
value: float = driver.applications.k70Vsa.sense.ddemod.search.burst.tolerance.
↳get()
```

This command controls burst search tolerance.

return

tolerance: Range: 1 to 15000, Unit: SYM

set(tolerance: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:BURSt:TOLerance
driver.applications.k70Vsa.sense.ddemod.search.burst.tolerance.set(tolerance =
↳1.0)
```

This command controls burst search tolerance.

param tolerance

Range: 1 to 15000, Unit: SYM

6.1.7.9.2.127 Mburst

class MburstCls

Mburst commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.mburst.clone()
```

Subgroups

6.1.7.9.2.128 Calc

SCPI Commands

```
SENSe:DDEMod:SEARch:MBURst:CALC
```

class CalcCls

Calc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:MBURst:CALC
value: float = driver.applications.k70Vsa.sense.ddemod.search.mburst.calc.get()
```

Sets the result range to be displayed after a single sweep (e.g. a burst number) .

return

select_res_range_nr: No help available

set(select_res_range_nr: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:MBURst:CALC
driver.applications.k70Vsa.sense.ddemod.search.mburst.calc.set(select_res_range_
↪nr = 1.0)
```

Sets the result range to be displayed after a single sweep (e.g. a burst number) .

param select_res_range_nr

No help available

6.1.7.9.2.129 Start

class StartCls

Start commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.mburst.start.clone()
```

Subgroups

6.1.7.9.2.130 Samples

SCPI Commands

```
SENSe:DDEMod:SEARch:MBURst:STARt:SAMPles
```

class SamplesCls

Samples commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:DDEMod:SEARch:MBURst:STARt:SAMPles
value: int = driver.applications.k70Vsa.sense.ddemod.search.mburst.start.
↳ samples.get()
```

This command queries the start sample of the current result range within the capture buffer. Tip: to query the start symbol, use [SENSe:]DDEMod:SEARch:MBURst:STARt[:SYMBols]?

return

start: No help available

6.1.7.9.2.131 Symbols

SCPI Commands

```
SENSe:DDEMod:SEARch:MBURst:STARt:SYMBols
```

class SymbolsCls

Symbols commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:DDEMod:SEARch:MBURst:STARt[:SYMBols]
value: int = driver.applications.k70Vsa.sense.ddemod.search.mburst.start.
↳ symbols.get()
```

This command queries the start symbol of the current result range within the capture buffer. Tip: to query the start sample, use [SENSe:]DDEMod:SEARch:MBURst:STARt:SAMPles?.

return

start: No help available

6.1.7.9.2.132 Pattern

class PatternCls

Pattern commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.pattern.clone()
```

Subgroups

6.1.7.9.2.133 Configure

class ConfigureCls

Configure commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.pattern.configure.clone()
```

Subgroups

6.1.7.9.2.134 Auto

SCPI Commands

```
SENSe:DDEMod:SEARch:PATtern:CONFigure:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARch:PATtern:CONFigure:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.search.pattern.configure.
↳ auto.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:PATtern:CONFigure:AUTO
driver.applications.k70Vsa.sense.ddemod.search.pattern.configure.auto.set(state,
↳ False)
```

No command help available

param state

No help available

6.1.7.9.2.135 Sync

class SyncCls

Sync commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.pattern.sync.clone()
```

Subgroups

6.1.7.9.2.136 Auto

SCPI Commands

```
SENSe:DDEMod:SEARCh:PATtern:SYNC:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:DDEMod:SEARCh:PATtern:SYNC:AUTO
value: enums.AutoManualMode = driver.applications.k70Vsa.sense.ddemod.search.
↳ pattern.sync.auto.get()
```

This command selects manual or automatic synchronization with a pattern waveform to speed up measurements.

return

use_wfm_for_sync: AUTO | MANual

set(use_wfm_for_sync: AutoManualMode) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:PATtern:SYNC:AUTO
driver.applications.k70Vsa.sense.ddemod.search.pattern.sync.auto.set(use_wfm_
↳ for_sync = enums.AutoManualMode.AUTO)
```

This command selects manual or automatic synchronization with a pattern waveform to speed up measurements.

param use_wfm_for_sync

AUTO | MANual

6.1.7.9.2.137 State

SCPI Commands

`SENSe:DDEMod:SEARCh:PATtern:SYNC:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARCh:PATtern:SYNC[:STATe]
value: bool = driver.applications.k70Vsa.sense.ddemod.search.pattern.sync.state.
↳get()
```

This command switches fast synchronization on and off, if you manually synchronize with a waveform pattern.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:PATtern:SYNC[:STATe]
driver.applications.k70Vsa.sense.ddemod.search.pattern.sync.state.set(state =
↳False)
```

This command switches fast synchronization on and off, if you manually synchronize with a waveform pattern.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.138 Pulse

class PulseCls

Pulse commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.pulse.clone()
```

Subgroups

6.1.7.9.2.139 State

SCPI Commands

```
SENSe:DDEMod:SEARch:PULSe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARch:PULSe:STATe
value: bool = driver.applications.k70Vsa.sense.ddemod.search.pulse.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:PULSe:STATe
driver.applications.k70Vsa.sense.ddemod.search.pulse.state.set(state = False)
```

No command help available

param state

No help available

6.1.7.9.2.140 Sync

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:COPY
SENSe:DDEMod:SEARch:SYNC:DELeTe
```

class SyncCls

Sync commands group definition. 18 total commands, 14 Subgroups, 2 group commands

copy(pattern: str) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:COPY
driver.applications.k70Vsa.sense.ddemod.search.sync.copy(pattern = '1')
```

This command copies a pattern file. The pattern to be copied must have been selected before using [SENSe:]DDEMod:SEARch:SYNC:NAME. Tip: In manual operation, a pattern can be copied in the editor by storing it under a new name.

param pattern

No help available

delete() → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:DELeTe
driver.applications.k70Vsa.sense.ddemod.search.sync.delete()
```

This command deletes a sync sequence. The sync sequence to be deleted must have been selected before using [SENSe]:DDEMod:SEARch:SYNC:NAME.

delete_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:DELeTe
driver.applications.k70Vsa.sense.ddemod.search.sync.delete_with_opc()
```

This command deletes a sync sequence. The sync sequence to be deleted must have been selected before using [SENSe]:DDEMod:SEARch:SYNC:NAME.

Same as delete, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.sync.clone()
```

Subgroups

6.1.7.9.2.141 Auto

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:AUTO
value: enums.AutoManualMode = driver.applications.k70Vsa.sense.ddemod.search.
↳sync.auto.get()
```

This command links the pattern search to the type of signal. When a signal is marked as patterned, pattern search is switched on automatically.

return

auto_patt_search: AUTO | MANual

set(auto_patt_search: AutoManualMode) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:AUTO
driver.applications.k70Vsa.sense.ddemod.search.sync.auto.set(auto_patt_search =
↳enums.AutoManualMode.AUTO)
```

This command links the pattern search to the type of signal. When a signal is marked as patterned, pattern search is switched on automatically.

param auto_patt_search
 AUTO | MANual

6.1.7.9.2.142 Catalog

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SelectionRangeB

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:CATalog
value: enums.SelectionRangeB = driver.applications.k70Vsa.sense.ddemod.search.
↳sync.catalog.get()
```

This command reads the names of all patterns stored on the hard disk. The file names are returned as a comma-separated list of strings, one for each file name (without the file extension) .

return
 patterns: CURRENT | ALL CURRENT Only patterns that belong to the current standard
 ALL All patterns

set(patterns: SelectionRangeB) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:CATalog
driver.applications.k70Vsa.sense.ddemod.search.sync.catalog.set(patterns =
↳enums.SelectionRangeB.ALL)
```

This command reads the names of all patterns stored on the hard disk. The file names are returned as a comma-separated list of strings, one for each file name (without the file extension) .

param patterns
 CURRENT | ALL CURRENT Only patterns that belong to the current standard ALL All
 patterns

6.1.7.9.2.143 Comment

SCPI Commands

SENSe:DDemod:SEARch:SYNC:COMMeNt

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:COMMeNt
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.comment.get()
```

This command defines a comment to a sync pattern. The pattern must have been selected before using [SENSe:]DDemod:SEARch:SYNC:NAME.

return
comment: No help available

set(comment: str) → None

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:COMMeNt
driver.applications.k70Vsa.sense.ddemod.search.sync.comment.set(comment = '1')
```

This command defines a comment to a sync pattern. The pattern must have been selected before using [SENSe:]DDemod:SEARch:SYNC:NAME.

param comment
No help available

6.1.7.9.2.144 Data

SCPI Commands

SENSe:DDemod:SEARch:SYNC:DATA

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:DATA
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.data.get()
```

This command defines the sync sequence of a sync pattern. The pattern must have been selected before using [SENSe:]DDemod:SEARch:SYNC:NAME. Important: The value range of a symbol depends on the degree of modulation, e.g. for an 8PSK modulation the value range is from 0 to 7. The degree of modulation belongs to the pattern and is set using the DDEM:SEAR:SYNC:NST command (see [SENSe:]DDemod:SEARch:SYNC:NSTate).

return
data: Four values represent a symbol (hexadecimal format). The value range of a symbol depends on the degree of modulation. With a degree of modulation of 4, all

symbols have a value range of: 0000, 0001, 0002, 0003 With a degree of modulation of 8: 0000, 0001, 0002, 0003, 0004, 0005, 0006, 0007

set(data: str) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:DATA
driver.applications.k70Vsa.sense.ddemod.search.sync.data.set(data = '1')
```

This command defines the sync sequence of a sync pattern. The pattern must have been selected before using [SENSe:]DDEMod:SEARch:SYNC:NAME. Important: The value range of a symbol depends on the degree of modulation, e.g. for an 8PSK modulation the value range is from 0 to 7. The degree of modulation belongs to the pattern and is set using the DDEMod:SEAR:SYNC:NST command (see [SENSe:]DDEMod:SEARch:SYNC:NState).

param data

Four values represent a symbol (hexadecimal format). The value range of a symbol depends on the degree of modulation. With a degree of modulation of 4, all symbols have a value range of: 0000, 0001, 0002, 0003 With a degree of modulation of 8: 0000, 0001, 0002, 0003, 0004, 0005, 0006, 0007

6.1.7.9.2.145 Found

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:FOUND
```

class FoundCls

Found commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:FOUND
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.found.get()
```

No command help available

return

name: No help available

6.1.7.9.2.146 IqcThreshold

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:IQThreshold
```

class IqcThresholdCls

IqcThreshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:IQThreshold
value: float = driver.applications.k70Vsa.sense.ddemod.search.sync.iqcThreshold.
    ↪get()
```

This command sets the I/Q correlation threshold for pattern matching in percent. A high level means stricter matching.

return

correlation_lev: Range: 10.0 to 100.0, Unit: PCT

set(correlation_lev: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:IQThreshold
driver.applications.k70Vsa.sense.ddemod.search.sync.iqcThreshold.
↪set(correlation_lev = 1.0)
```

This command sets the I/Q correlation threshold for pattern matching in percent. A high level means stricter matching.

param correlation_lev

Range: 10.0 to 100.0, Unit: PCT

6.1.7.9.2.147 Mode

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SyncMode

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:MODE
value: enums.SyncMode = driver.applications.k70Vsa.sense.ddemod.search.sync.
↪mode.get()
```

This command sets the vector analyzer so that the measurement is performed only if the measurement was synchronous to the selected sync pattern. The command is available only if the pattern search is activated (see [SENSe:]DDEMod:SEARch:SYNC:STATe) .

return

meas_only_on_patt: MEAS | SYNC MEAS The measurement is performed independently of successful synchronization SYNC The measured values are displayed and considered in the error evaluation only if the set sync pattern was found. Bursts with a wrong sync pattern (sync not found) are ignored. If an invalid or no sync pattern is found, the measurement waits and resumes running only when a valid sync pattern is found.

set(meas_only_on_patt: SyncMode) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:MODE
driver.applications.k70Vsa.sense.ddemod.search.sync.mode.set(meas_only_on_patt,
↪enums.SyncMode.MEAS)
```

This command sets the vector analyzer so that the measurement is performed only if the measurement was synchronous to the selected sync pattern. The command is available only if the pattern search is activated (see [SENSe:]DDEMod:SEARch:SYNC:STATe) .

param meas_only_on_patt

MEAS | SYNC MEAS The measurement is performed independently of successful synchronization SYNC The measured values are displayed and considered in the error evaluation only if the set sync pattern was found. Bursts with a wrong sync pattern (sync not found) are ignored. If an invalid or no sync pattern is found, the measurement waits and resumes running only when a valid sync pattern is found.

6.1.7.9.2.148 Name**SCPI Commands**

```
SENSe:DDemod:SEARch:SYNC:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:NAME
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.name.get()
```

No command help available

return

name: No help available

set(name: str) → None

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:NAME
driver.applications.k70Vsa.sense.ddemod.search.sync.name.set(name = '1')
```

No command help available

param name

No help available

6.1.7.9.2.149 Nstate**SCPI Commands**

```
SENSe:DDemod:SEARch:SYNC:NState
```

class NstateCls

Nstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDemod:SEARch:SYNC:NState
value: float = driver.applications.k70Vsa.sense.ddemod.search.sync.nstate.get()
```

This command selects the degree of modulation (number of permitted states) . The pattern must have been selected before using [SENSe:]DDemod:SEARch:SYNC:NAME. The number of permitted states depends on the modulation mode.

return

nstate: No help available

set(nstate: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:NState
driver.applications.k70Vsa.sense.ddemod.search.sync.nstate.set(nstate = 1.0)
```

This command selects the degree of modulation (number of permitted states) . The pattern must have been selected before using [SENSe:]DDEMod:SEARch:SYNC:NAME. The number of permitted states depends on the modulation mode.

param nstate

No help available

6.1.7.9.2.150 Offset

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:OFFSet
value: float = driver.applications.k70Vsa.sense.ddemod.search.sync.offset.get()
```

No command help available

return

arg_0: No help available

set(arg_0: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:OFFSet
driver.applications.k70Vsa.sense.ddemod.search.sync.offset.set(arg_0 = 1.0)
```

No command help available

param arg_0

No help available

6.1.7.9.2.151 Pattern

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:PATtern
```

class PatternCls

Pattern commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:PATtern
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.pattern.get()
```

No command help available

```
return
arg_0: No help available
```

set(arg_0: str) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:PATtern
driver.applications.k70Vsa.sense.ddemod.search.sync.pattern.set(arg_0 = '1')
```

No command help available

```
param arg_0
No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.search.sync.pattern.clone()
```

Subgroups

6.1.7.9.2.152 Add

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:PATtern:ADD
```

class AddCls

Add commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(add_pattern: str) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:PATtern:ADD
driver.applications.k70Vsa.sense.ddemod.search.sync.pattern.add.set(add_pattern_
↪ = '1')
```

This command adds a pattern to the current standard. Using the DDEMod:SEAR:SYNC:SEL command, only those patterns can be selected which belong to the current standard (see [SENSe:]DDEMod:SEARch:SYNC:SElect).

```
param add_pattern
No help available
```

6.1.7.9.2.153 Remove

SCPI Commands

SENSe:DDEMod:SEARch:SYNC:PATtern:REMove

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*pattern: SelectAll*) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:PATtern:REMove
driver.applications.k70Vsa.sense.ddemod.search.sync.pattern.remove.set(pattern_
↳= enums.SelectAll.ALL)
```

This command deletes one or all patterns from the current standard.

param pattern

(enum or string) No help available

6.1.7.9.2.154 Select

SCPI Commands

SENSe:DDEMod:SEARch:SYNC:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:SElect
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.select.get()
```

This command selects a predefined sync pattern file.

return

filename: No help available

set(*filename: str*) → None

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC:SElect
driver.applications.k70Vsa.sense.ddemod.search.sync.select.set(filename = '1')
```

This command selects a predefined sync pattern file.

param filename

No help available

6.1.7.9.2.155 State

SCPI Commands

```
SENSe:DDEMod:SEARCh:SYNC:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARCh:SYNC:STATe
value: bool = driver.applications.k70Vsa.sense.ddemod.search.sync.state.get()
```

This command switches the search for a sync sequence on or off.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:SYNC:STATe
driver.applications.k70Vsa.sense.ddemod.search.sync.state.set(state = False)
```

This command switches the search for a sync sequence on or off.

param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.156 Text

SCPI Commands

```
SENSe:DDEMod:SEARCh:SYNC:TEXT
```

class TextCls

Text commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:SEARCh:SYNC:TEXT
value: str = driver.applications.k70Vsa.sense.ddemod.search.sync.text.get()
```

This command defines a text to explain the pattern. The text is displayed only in the selection menu (manual control) . This text should be short and concise. Detailed information about the pattern is given in the comment (see [SENSe:]DDEMod:SEARCh:SYNC:COMMENT) .

return
text: No help available

set(text: str) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:SYNC:TEXT
driver.applications.k70Vsa.sense.ddemod.search.sync.text.set(text = '1')
```

This command defines a text to explain the pattern. The text is displayed only in the selection menu (manual control) . This text should be short and concise. Detailed information about the pattern is given in the comment (see [SENSe:]DDEMod:SEARCh:SYNc:COMMeNt) .

param text

No help available

6.1.7.9.2.157 Time

SCPI Commands

`SENSe:DDEMod:SEARCh:TIME`

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SEARCh:TIME
value: float = driver.applications.k70Vsa.sense.ddemod.search.time.get()
```

No command help available

return

time: No help available

set(time: float) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:TIME
driver.applications.k70Vsa.sense.ddemod.search.time.set(time = 1.0)
```

No command help available

param time

No help available

6.1.7.9.2.158 Signal

class SignalCls

Signal commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.signal.clone()
```


Subgroups

6.1.7.9.2.159 Pattern

SCPI Commands

```
SENSe:DDemod:SIGNal:PATtern
```

class PatternCls

Pattern commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDemod:SIGNal:PATtern
value: bool = driver.applications.k70Vsa.sense.ddemod.signal.pattern.get()
```

This command specifies whether the signal contains a pattern or not.

return

state: ON | OFF | 0 | 1 OFF | 0 The signal does not contain a pattern. ON | 1 The signal contains a pattern.

set(state: bool) → None

```
# SCPI: [SENSe]:DDemod:SIGNal:PATtern
driver.applications.k70Vsa.sense.ddemod.signal.pattern.set(state = False)
```

This command specifies whether the signal contains a pattern or not.

param state

ON | OFF | 0 | 1 OFF | 0 The signal does not contain a pattern. ON | 1 The signal contains a pattern.

6.1.7.9.2.160 Value

SCPI Commands

```
SENSe:DDemod:SIGNal:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DdemSignalType

```
# SCPI: [SENSe]:DDemod:SIGNal[:VALue]
value: enums.DdemSignalType = driver.applications.k70Vsa.sense.ddemod.signal.
    ↪ value.get()
```

No command help available

return

signal_type: CONTinuous | BURSted

set(*signal_type: DdemSignalType*) → None

```
# SCPI: [SENSe]:DDEMod:SIGNal[:VALue]
driver.applications.k70Vsa.sense.ddemod.signal.value.set(signal_type = enums.
↳ DdemSignalType.BURSted)
```

No command help available

param signal_type
CONTInuous | BURSted

6.1.7.9.2.161 Standard

SCPI Commands

```
SENSe:DDEMod:STANdard:SAVE
SENSe:DDEMod:STANdard:DELeTe
```

class StandardCls

Standard commands group definition. 6 total commands, 3 Subgroups, 2 group commands

delete(*filename: str*) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:DELeTe
driver.applications.k70Vsa.sense.ddemod.standard.delete(filename = '1')
```

This command deletes a specified digital standard file in the vector signal analysis.

param filename
File name including the path for the digital standard file

save(*filename: str*) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:SAVE
driver.applications.k70Vsa.sense.ddemod.standard.save(filename = '1')
```

This command stores the current settings of the vector signal analysis as a new user-defined digital standard. If the name of the digital standard is already in use, an error message is output and a new name has to be selected. It is recommended that you define a comment before storing the standard.

param filename
The path and file name to which the settings are stored.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.standard.clone()
```

Subgroups

6.1.7.9.2.162 Comment

SCPI Commands

SENSe:DDEMod:STANdard:COMMeNt

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:STANdard:COMMeNt
value: str = driver.applications.k70Vsa.sense.ddemod.standard.comment.get()
```

This command enters the comment for a new standard. The comment is stored with the standard and is only displayed in the selection menu (manual operation) . In remote control, the string is deleted after the standard has been stored, allowing a new comment to be entered for the next standard. In this case a blank string is returned when for the query.

return

comment: No help available

set(comment: str) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:COMMeNt
driver.applications.k70Vsa.sense.ddemod.standard.comment.set(comment = '1')
```

This command enters the comment for a new standard. The comment is stored with the standard and is only displayed in the selection menu (manual operation) . In remote control, the string is deleted after the standard has been stored, allowing a new comment to be entered for the next standard. In this case a blank string is returned when for the query.

param comment

No help available

6.1.7.9.2.163 Preset

class PresetCls

Preset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.standard.preset.clone()
```

Subgroups

6.1.7.9.2.164 Value

SCPI Commands

`SENSe:DDEMod:STANdard:PRESet:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:DDEMod:STANdard:PRESet[:VALue]
driver.applications.k70Vsa.sense.ddemod.standard.preset.value.set()
```

This command restores the default settings of the currently selected standard.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:PRESet[:VALue]
driver.applications.k70Vsa.sense.ddemod.standard.preset.value.set_with_opc()
```

This command restores the default settings of the currently selected standard.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.1.7.9.2.165 Sync

class SyncCls

Sync commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.standard.sync.clone()
```

Subgroups

6.1.7.9.2.166 Offset

class OffsetCls

Offset commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.standard.sync.offset.clone()
```

Subgroups

6.1.7.9.2.167 State

SCPI Commands

```
SENSe:DDEMod:STANdard:SYNC:OFFSet:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:STANdard:SYNC:OFFSet:STATe
value: bool = driver.applications.k70Vsa.sense.ddemod.standard.sync.offset.
↳state.get()
```

This command (de) activates the pattern offset.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:SYNC:OFFSet:STATe
driver.applications.k70Vsa.sense.ddemod.standard.sync.offset.state.set(state =
↳False)
```

This command (de) activates the pattern offset.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.9.2.168 Value

SCPI Commands

```
SENSe:DDEMod:STANdard:SYNC:OFFSet:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:STANdard:SYNc:OFFSet[:VALue]
value: float = driver.applications.k70Vsa.sense.ddemod.standard.sync.offset.
↳ value.get()
```

This command defines a number of symbols which are ignored before the comparison with the pattern starts.

return

offset: Range: 0 to 15000, Unit: SYMB

set(offset: float) → None

```
# SCPI: [SENSe]:DDEMod:STANdard:SYNc:OFFSet[:VALue]
driver.applications.k70Vsa.sense.ddemod.standard.sync.offset.value.set(offset =
↳ 1.0)
```

This command defines a number of symbols which are ignored before the comparison with the pattern starts.

param offset

Range: 0 to 15000, Unit: SYMB

6.1.7.9.2.169 SymbolRate

SCPI Commands

```
SENSe:DDEMod:SRATe
```

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:SRATe
value: float = driver.applications.k70Vsa.sense.ddemod.symbolRate.get()
```

This command defines the symbol rate. The minimum symbol rate is 25 Hz. The maximum symbol rate depends on the defined ‘Sample Rate’ (see ‘Sample rate, symbol rate and I/Q bandwidth’).

return

symbol_rate: Range: 25 to 250e6, Unit: HZ

set(symbol_rate: float) → None

```
# SCPI: [SENSe]:DDEMod:SRATe
driver.applications.k70Vsa.sense.ddemod.symbolRate.set(symbol_rate = 1.0)
```

This command defines the symbol rate. The minimum symbol rate is 25 Hz. The maximum symbol rate depends on the defined ‘Sample Rate’ (see ‘Sample rate, symbol rate and I/Q bandwidth’).

param symbol_rate

Range: 25 to 250e6, Unit: HZ

6.1.7.9.2.170 Tfilter

class TfilterCls

Tfilter commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.tfilter.clone()
```

Subgroups

6.1.7.9.2.171 Alpha

SCPI Commands

```
SENSe:DDEMod:TFILter:ALPHA
```

class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:DDEMod:TFILter:ALPHA
value: float = driver.applications.k70Vsa.sense.ddemod.tfilter.alpha.get()
```

This command determines the TX filter characteristic (ALPHA/BT) .

return

alpha: Range: 0.03 to 1.0

set(alpha: float) → None

```
# SCPI: [SENSe]:DDEMod:TFILter:ALPHA
driver.applications.k70Vsa.sense.ddemod.tfilter.alpha.set(alpha = 1.0)
```

This command determines the TX filter characteristic (ALPHA/BT) .

param alpha

Range: 0.03 to 1.0

6.1.7.9.2.172 Name

SCPI Commands

```
SENSe:DDEMod:TFILter:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:TFILter:NAME
value: str = driver.applications.k70Vsa.sense.ddemod.tfilter.name.get()
```

This command selects a transmit filter and automatically switches it on. For more information on transmit filters, refer to ‘Transmit filters’.

return

name: string Name of the Transmit filter; an overview of available transmit filters is provided in ‘Transmit filters’.

set(name: str) → None

```
# SCPI: [SENSe]:DDEMod:TFILter:NAME
driver.applications.k70Vsa.sense.ddemod.tfilter.name.set(name = '1')
```

This command selects a transmit filter and automatically switches it on. For more information on transmit filters, refer to ‘Transmit filters’.

param name

string Name of the Transmit filter; an overview of available transmit filters is provided in ‘Transmit filters’.

6.1.7.9.2.173 State

SCPI Commands

SENSe:DDEMod:TFILter:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:TFILter[:STATe]
value: bool = driver.applications.k70Vsa.sense.ddemod.tfilter.state.get()
```

Use this command to switch the transmit filter off. To switch a transmit filter on, use the [SENSe:]DDEMod:TFILter:NAME command.

return

state: OFF | 0 Switches the transmit filter off. ON | 1 Switches the transmit filter specified by [SENSe:]DDEMod:TFILter:NAME on. However, this command is not necessary, as the [SENSe:]DDEMod:TFILter:NAME command automatically switches the filter on.

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:TFILter[:STATe]
driver.applications.k70Vsa.sense.ddemod.tfilter.state.set(state = False)
```

Use this command to switch the transmit filter off. To switch a transmit filter on, use the [SENSe:]DDEMod:TFILter:NAME command.

param state

OFF | 0 Switches the transmit filter off. ON | 1 Switches the transmit filter specified by [SENSe:]DDEMod:TFILter:NAME on. However, this command is not necessary, as the [SENSe:]DDEMod:TFILter:NAME command automatically switches the filter on.

6.1.7.9.2.174 User**SCPI Commands**

SENSe:DDEMod:TFILter:USER

class UserCls

User commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

<pre># SCPI: [SENSe]:DDEMod:TFILter:USER value: str = driver.applications.k70Vsa.sense.ddemod.tfilter.user.get()</pre>
--

This command selects a user-defined transmit filter file.

return

filter_name: The name of the transmit filter file.

set(filter_name: str) → None

<pre># SCPI: [SENSe]:DDEMod:TFILter:USER driver.applications.k70Vsa.sense.ddemod.tfilter.user.set(filter_name = '1')</pre>
--

This command selects a user-defined transmit filter file.

param filter_name

The name of the transmit filter file.

6.1.7.9.2.175 Time**SCPI Commands**

SENSe:DDEMod:TIME

class TimeCls

Time commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

<pre># SCPI: [SENSe]:DDEMod:TIME value: float = driver.applications.k70Vsa.sense.ddemod.time.get()</pre>
--

The command determines the number of displayed symbols (result length) .

return

result_length: numeric value Range: 10 to 64000, Unit: Sym

set(*result_length: float*) → None

```
# SCPI: [SENSe]:DDEMod:TIME
driver.applications.k70Vsa.sense.ddemod.time.set(result_length = 1.0)
```

The command determines the number of displayed symbols (result length) .

param result_length

numeric value Range: 10 to 64000, Unit: Sym

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.time.clone()
```

Subgroups

6.1.7.9.2.176 Auto

SCPI Commands

```
SENSe:DDEMod:TIME:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:TIME:AUTO
value: bool = driver.applications.k70Vsa.sense.ddemod.time.auto.get()
```

Determines how the result length is defined for multi-modulation analysis. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

return

state: ON | OFF | 0 | 1 OFF | 0 The result length is specified by [SENSe:]DDEMod:TIME. ON | 1 The result length is set to the number defined in the currently loaded Frame Structure file.

set(*state: bool*) → None

```
# SCPI: [SENSe]:DDEMod:TIME:AUTO
driver.applications.k70Vsa.sense.ddemod.time.auto.set(state = False)
```

Determines how the result length is defined for multi-modulation analysis. This command is only available if the additional Multi-Modulation Analysis option (R&S FSWP-K70M) is installed.

param state

ON | OFF | 0 | 1 OFF | 0 The result length is specified by [SENSe:]DDEMod:TIME. ON | 1 The result length is set to the number defined in the currently loaded Frame Structure file.

6.1.7.9.2.177 User

class UserCls

User commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.ddemod.user.clone()
```

Subgroups

6.1.7.9.2.178 Name

SCPI Commands

```
SENSe:DDEMod:USER:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:DDEMod:USER:NAME
value: str = driver.applications.k70Vsa.sense.ddemod.user.name.get()
```

Selects the file that contains the user-defined modulation to be loaded.

return

filename: Path and file name of the *.vam file The default storage location for user-defined modulations is C:/R_S/INSTR/USER/vsa/Constellation.

set(filename: str) → None

```
# SCPI: [SENSe]:DDEMod:USER:NAME
driver.applications.k70Vsa.sense.ddemod.user.name.set(filename = '1')
```

Selects the file that contains the user-defined modulation to be loaded.

param filename

Path and file name of the *.vam file The default storage location for user-defined modulations is C:/R_S/INSTR/USER/vsa/Constellation.

6.1.7.9.3 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.frequency.clone()
```

Subgroups

6.1.7.9.3.1 Center

class CenterCls

Center commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.frequency.center.clone()
```

Subgroups

6.1.7.9.3.2 Step

class StepCls

Step commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.frequency.center.step.clone()
```

Subgroups

6.1.7.9.3.3 Auto

SCPI Commands

```
SENSe:FREQuency:CENTer:STEP:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:AUTO
value: bool = driver.applications.k70Vsa.sense.frequency.center.step.auto.get()
```

Defines the step width of the center frequency.

return

state: ON | 1 Links the step width to the current standard (currently 1 MHz for all standards) OFF | 0 Sets the step width as defined using the `FREQ:CENT:STEP` command (see `[SENSe]:FREQuency:CENTer:STEP`).

set(state: bool) → None

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:AUTO
driver.applications.k70Vsa.sense.frequency.center.step.auto.set(state = False)
```

Defines the step width of the center frequency.

param state

ON | 1 Links the step width to the current standard (currently 1 MHz for all standards) OFF | 0 Sets the step width as defined using the `FREQ:CENT:STEP` command (see `[SENSe]:FREQuency:CENTer:STEP`).

6.1.7.9.3.4 Offset

SCPI Commands

```
SENSe:FREQuency:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:OFFSet
value: float = driver.applications.k70Vsa.sense.frequency.offset.get()
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type 'frequency' will be corrected for this shift numerically by the application. See also 'Frequency Offset'. Note: In MSRA mode, the setting command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

return

frequency_offset: No help available

set(frequency_offset: float) → None

```
# SCPI: [SENSe]:FREQuency:OFFSet
driver.applications.k70Vsa.sense.frequency.offset.set(frequency_offset = 1.0)
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type 'frequency' will be corrected for this shift numerically by the application. See also 'Frequency Offset'. Note: In MSRA mode, the setting

command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

param frequency_offset

Range: -1 THz to 1 THz, Unit: HZ

6.1.7.9.4 Msra

class MsraCls

Msra commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.msra.clone()
```

Subgroups

6.1.7.9.4.1 Capture

class CaptureCls

Capture commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.msra.capture.clone()
```

Subgroups

6.1.7.9.4.2 Offset

SCPI Commands

```
SENSe:MSRA:CAPTure:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MSRA:CAPTure:OFFSet
value: float = driver.applications.k70Vsa.sense.msra.capture.offset.get()
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

return

offset: This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

set(offset: float) → None

```
# SCPI: [SENSe]:MSRA:CAPTure:OFFSet
driver.applications.k70Vsa.sense.msra.capture.offset.set(offset = 1.0)
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

param offset

This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

6.1.7.9.5 SwapIq

SCPI Commands

```
SENSe:SWAPiq
```

class SwapIqCls

SwapIq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWAPiq
value: bool = driver.applications.k70Vsa.sense.swapIq.get()
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

return

state: ON | 1 I and Q signals are interchanged Inverted sideband, Q+j*I OFF | 0 I and Q signals are not interchanged Normal sideband, I+j*Q

set(state: bool) → None

```
# SCPI: [SENSe]:SWAPiq
driver.applications.k70Vsa.sense.swapIq.set(state = False)
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

param state

ON | 1 I and Q signals are interchanged Inverted sideband, Q+j*I OFF | 0 I and Q signals are not interchanged Normal sideband, I+j*Q

6.1.7.9.6 Sweep

class SweepCls

Sweep commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.sweep.clone()
```

Subgroups

6.1.7.9.6.1 Count

class CountCls

Count commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.sweep.count.clone()
```

Subgroups

6.1.7.9.6.2 Current

SCPI Commands

```
SENSe:SWEEp:COUNT:CURRENT
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(counter: Optional[Counter] = None) → int

```
# SCPI: [SENSe]:SWEEp:COUNT:CURRENT
value: int = driver.applications.k70Vsa.sense.sweep.count.current.get(counter =
↳ enums.Counter.CAPTURE)
```

This command queries the current statistics counter value which indicates how many result ranges have been evaluated. For results that use the capture buffer as a source, the number of used capture buffers can be queried.

param counter

CAPTURE | STATistics STATistics Returns the number of result ranges that have been evaluated. CAPTURE Returns the number of used capture buffers evaluated.

return

count: No help available

6.1.7.9.6.3 Value

SCPI Commands

```
SENSe:SWEEp:COUNT:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEp:COUNT[:VALue]
value: float = driver.applications.k70Vsa.sense.sweep.count.value.get()
```

No command help available

return

sweep_count: No help available

set(sweep_count: float) → None

```
# SCPI: [SENSe]:SWEEp:COUNT[:VALue]
driver.applications.k70Vsa.sense.sweep.count.value.set(sweep_count = 1.0)
```

No command help available

param sweep_count

No help available

6.1.7.9.7 Tcapture

class TcaptureCls

Tcapture commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.sense.tcapture.clone()
```

Subgroups

6.1.7.9.7.1 Length

SCPI Commands

```
SENSe:TCAPture:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:TCAPture:LENGth
value: float = driver.applications.k70Vsa.sense.tcapture.length.get()
```

No command help available

return
arg_0: No help available

set(arg_0: float) → None

```
# SCPI: [SENSe]:TCAPture:LENGth
driver.applications.k70Vsa.sense.tcapture.length.set(arg_0 = 1.0)
```

No command help available

param arg_0
No help available

6.1.7.10 Status

class StatusCls

Status commands group definition. 35 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.clone()
```

Subgroups

6.1.7.10.1 Questionable

class QuestionableCls

Questionable commands group definition. 35 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.clone()
```

Subgroups

6.1.7.10.1.1 Modulation<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.status.questionable.modulation.repcap_window_get()
driver.applications.k70Vsa.status.questionable.modulation.repcap_window_set(repcap.
↳Window.Nr1)
```

class ModulationCls

Modulation commands group definition. 35 total commands, 11 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.clone()
```

Subgroups

6.1.7.10.1.2 Cfrequency

class CfrequencyCls

Cfrequency commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.cfrequency.clone()
```

Subgroups

6.1.7.10.1.3 Condition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:CFRequency:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → str`

```
# SCPI: STATus:QUEStionable:MODulation<Window>:CFRequency:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.
↳cfrequency.condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.4 Enable

SCPI Commands

```
STATUS:QUESTionable:MODulation<Window>:CFrequency:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATUS:QUESTionable:MODulation<Window>:CFrequency:ENABle
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
↳cfrequency.enable.get(window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATUS:QUESTionable:MODulation<Window>:CFrequency:ENABle
driver.applications.k70Vsa.status.questionable.modulation.cfrequency.enable.
↳set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.5 Event**SCPI Commands**

STATus:QUESTionable:MODulation<Window>:CFRequency:EVENT

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

<pre># SCPI: STATus:QUESTionable:MODulation<Window>:CFRequency[:EVENT] value: str = driver.applications.k70Vsa.status.questionable.modulation. cfrequency.event.get(window = repcap.Window.Default)</pre>

This command reads out the EVENT section of the status register. The command also deletes the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.6 Ntransition**SCPI Commands**

STATus:QUESTionable:MODulation<Window>:CFRequency:NTRansition

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → NtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:CFRequency:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    ↪ modulation.cfrequency.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:CFRequency:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.cfrequency.
    ↪ ntransition.set(bit_definition = 1, channel_name = '1', window = repcap.
    ↪ Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.7 Ptransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:CFRequency:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:MODulation<Window>:CFrequency:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    ↪ modulation.cfrequency.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATUS:QUESTIONABLE:MODulation<Window>:CFrequency:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.cfrequency.
    ↪ ptransition.set(bit_definition = 1, channel_name = '1', window = repcap.
    ↪ Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.8 Condition

SCPI Commands

```
STATUS:QUESTIONABLE:MODulation<Window>:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATUS:QUESTIONABLE:MODulation<Window>:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.
    ↪ condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.9 Enable**SCPI Commands**

```
STATus:QUESTionable:MODulation<Window>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:ENABle
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
enable.get(window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:ENABle
driver.applications.k70Vsa.status.questionable.modulation.enable.set(bit_
definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.10 Event**SCPI Commands**

```
STATus:QUESTionable:MODulation<Window>:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUESTionable:MODulation<Window>[:EVENT]
value: str = driver.applications.k70Vsa.status.questionable.modulation.event.
    ↪get(window = repcap.Window.Default)
```

This command reads out the EVENT section of the status register. The command also deletes the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.11 Evm**class EvmCls**

Evm commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.evm.clone()
```

Subgroups

6.1.7.10.1.12 Condition

SCPI Commands

STATUS:QUESTIONable:MODulation<Window>:EVM:CONDition

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATUS:QUESTIONable:MODulation<Window>:EVM:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.evm.
↳ condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.13 Enable

SCPI Commands

STATUS:QUESTIONable:MODulation<Window>:EVM:ENABLE

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATUS:QUESTIONable:MODulation<Window>:EVM:ENABLE
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
↳ evm.enable.get(window = repcap.Window.Default)
```

This command controls the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None, *window*=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:EVM:ENABle
driver.applications.k70Vsa.status.questionable.modulation.evm.enable.set(bit_
↳definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVEnt part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.14 Event**SCPI Commands**

```
STATus:QUEStionable:MODulation<Window>:EVM:EVEnt
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default) → str

```
# SCPI: STATus:QUEStionable:MODulation<Window>:EVM[:EVEnt]
value: str = driver.applications.k70Vsa.status.questionable.modulation.evm.
↳event.get(window = repcap.Window.Default)
```

This command reads out the EVEnt section of the status register. The command also deletes the contents of the EVEnt section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.15 Ntransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:EVM:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Bit_Definition**: int: Range: 0 to 65535
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(*window=Window.Default*) → NtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:EVM:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.evm.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None, window=Window.Default*) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:EVM:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.evm.ntransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.16 Ptransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:EVM:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:EVM:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.evm.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:EVM:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.evm.ptransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.17 Fsk

class FskCls

Fsk commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.fsk.clone()
```

Subgroups

6.1.7.10.1.18 Condition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:FSK:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.fsk.
    condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Modulation’)

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.19 Enable

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:FSK:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535

- **Channel_Name:** str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK:ENABle
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
↳ fsk.enable.get(window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVEnt part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK:ENABle
driver.applications.k70Vsa.status.questionable.modulation.fsk.enable.set(bit_
↳ definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVEnt part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.20 Event

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:FSK:EVEnt
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK[:EVENT]
value: str = driver.applications.k70Vsa.status.questionable.modulation.fsk.
↳event.get(window = repcap.Window.Default)
```

This command reads out the EVENT section of the status register. The command also deletes the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.21 Ntransition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:FSK:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → NtransitionStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
↳modulation.fsk.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:FSK:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.fsk.ntransition.
↳set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```


This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.22 Ptransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:FSK:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:FSK:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.fsk.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:FSK:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.fsk.ptransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.23 IqRho

class IqRhoCls

IqRho commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.iqRho.clone()
```

Subgroups

6.1.7.10.1.24 Condition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:IQRHo:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUESTionable:MODulation<Window>:IQRHo:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.iqRho.
    ↪condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.25 Enable

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:IQRHo:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:IQRHo:ENABle
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
↳iqRho.enable.get(window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:IQRHo:ENABle
driver.applications.k70Vsa.status.questionable.modulation.iqRho.enable.set(bit_
↳definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.26 Event

SCPI Commands

`STATus:QUESTionable:MODulation<Window>:IQRHo:EVENT`

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUESTionable:MODulation<Window>:IQRHo[:EVENT]
value: str = driver.applications.k70Vsa.status.questionable.modulation.iqRho.
    ↪event.get(window = repcap.Window.Default)
```

This command reads out the EVENT section of the status register. The command also deletes the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.27 Ntransition

SCPI Commands

`STATus:QUESTionable:MODulation<Window>:IQRHo:NTRansition`

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → NtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:IQRHo:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    ↪modulation.iqRho.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TTransition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:IQRHo:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.iqRho.ntransition.
↪set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.28 Ptransition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:IQRHo:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:IQRHo:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
↪modulation.iqRho.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None, *window*=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:IQRHo:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.iqRho.ptransition.
↪set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.29 Magnitude

class MagnitudeCls

Magnitude commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.magnitude.clone()
```

Subgroups

6.1.7.10.1.30 Condition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:MAGNitude:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default) → str

```
# SCPI: STATus:QUEStionable:MODulation<Window>:MAGNitude:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.
↪magnitude.condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.31 Enable**SCPI Commands**

```
STaTus:QUEStionable:MODulation<Window>:MAGNitude:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=*Window.Default*) → EnableStruct

```
# SCPI: STaTus:QUEStionable:MODulation<Window>:MAGNitude:ENABle
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
    ↪magnitude.enable.get(window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENt part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=*Window.Default*) → None

```
# SCPI: STaTus:QUEStionable:MODulation<Window>:MAGNitude:ENABle
driver.applications.k70Vsa.status.questionable.modulation.magnitude.enable.
    ↪set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENt part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.32 Event

SCPI Commands

`STATus:QUESTionable:MODulation<Window>:MAGNitude:EVENTt`

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*) → str

```
# SCPI: STATus:QUESTionable:MODulation<Window>:MAGNitude[:EVENTt]
value: str = driver.applications.k70Vsa.status.questionable.modulation.
    ↪magnitude.event.get(window = repcap.Window.Default)
```

This command reads out the EVENTt section of the status register. The command also deletes the contents of the EVENTt section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.33 Ntransition

SCPI Commands

`STATus:QUESTionable:MODulation<Window>:MAGNitude:NTRansition`

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=*Window.Default*) → NtransitionStruct


```
# SCPI: STATus:QUEStionable:MODulation<Window>:MAGNitude:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    ↪ modulation.magnitude.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:MAGNitude:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.magnitude.ntransition.
    ↪ set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.34 Ptransition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:MAGNitude:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:MAGNitude:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.magnitude.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:MAGNitude:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.magnitude.ptransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.35 Ntransition

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → NtransitionStruct

```
# SCPI: STATus:QUEStionable:MODulation<Window>:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    ↪ modulation.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None, window=Window.Default*) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.ntransition.set(bit_
    ↪ definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.36 Phase

class PhaseCls

Phase commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.status.questionable.modulation.phase.clone()
```

Subgroups

6.1.7.10.137 Condition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:PHASe:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:CONDition
value: str = driver.applications.k70Vsa.status.questionable.modulation.phase.
↳ condition.get(window = repcap.Window.Default)
```

This command reads out the CONDition section of the status register. The command does not delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.138 Enable

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:PHASe:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:ENABLE
value: EnableStruct = driver.applications.k70Vsa.status.questionable.modulation.
↳ phase.enable.get(window = repcap.Window.Default)
```

This command controls the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None, *window*=Window.Default) → None

```
# SCPI: STATus:QUEStionable:MODulation<Window>:PHASe:ENABle
driver.applications.k70Vsa.status.questionable.modulation.phase.enable.set(bit_
↳definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.39 Event

SCPI Commands

```
STATus:QUEStionable:MODulation<Window>:PHASe:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default) → str

```
# SCPI: STATus:QUEStionable:MODulation<Window>:PHASe[:EVENT]
value: str = driver.applications.k70Vsa.status.questionable.modulation.phase.
↳event.get(window = repcap.Window.Default)
```

This command reads out the EVENT section of the status register. The command also deletes the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.1.7.10.1.40 Ntransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:PHASe:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Bit_Definition**: int: Range: 0 to 65535
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(*window=Window.Default*) → NtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:NTRansition
value: NtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.phase.ntransition.get(window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None, window=Window.Default*) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:NTRansition
driver.applications.k70Vsa.status.questionable.modulation.phase.ntransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

This command controls the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.41 Ptransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:PHASe:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: Range: 0 to 65535
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.phase.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PHASe:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.phase.ptransition.
    set(bit_definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.10.1.42 Ptransition

SCPI Commands

```
STATus:QUESTionable:MODulation<Window>:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Bit_Definition**: int: Range: 0 to 65535
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(*window=Window.Default*) → PtransitionStruct

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PTRansition
value: PtransitionStruct = driver.applications.k70Vsa.status.questionable.
    modulation.ptransition.get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None, window=Window.Default*) → None

```
# SCPI: STATus:QUESTionable:MODulation<Window>:PTRansition
driver.applications.k70Vsa.status.questionable.modulation.ptransition.set(bit_
    definition = 1, channel_name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param bit_definition

Range: 0 to 65535

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Modulation')

6.1.7.11 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.applications.k70Vsa.trace.repcap_window_get()
driver.applications.k70Vsa.trace.repcap_window_set(repcap.Window.Nr1)
```

class TraceCls

Trace commands group definition. 5 total commands, 2 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trace.clone()
```

Subgroups

6.1.7.11.1 Data

SCPI Commands

```
FORMAT REAL, 32; TRACe<n>[:DATA]
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_type: TraceTypeDdem, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.applications.k70Vsa.trace.data.get(trace_type = enums.TraceTypeDdem.MSTRace, window = repcap.Window.Default)
```

This command queries the trace data. Which data is returned depends on the result display in the window specified by the suffix <n>. For details see ‘Measurement results for TRACe<n>[:DATA]? TRACe<n>’.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trace’)

return

trace_ydata: No help available

6.1.7.11.2 Iq

class IqCls

Iq commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trace.iq.clone()
```

Subgroups

6.1.7.11.2.1 Bandwidth

SCPI Commands

```
TRACe:IQ:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:BWIDth
value: float = driver.applications.k70Vsa.trace.iq.bandwidth.get()
```

This command queries the bandwidth in Hz of the resampling filter ('Usable I/Q Bandwidth').

return

bandwidth: Usable I/Q bandwidth Unit: Hz

set(bandwidth: float) → None

```
# SCPI: TRACe:IQ:BWIDth
driver.applications.k70Vsa.trace.iq.bandwidth.set(bandwidth = 1.0)
```

This command queries the bandwidth in Hz of the resampling filter ('Usable I/Q Bandwidth').

param bandwidth

Usable I/Q bandwidth Unit: Hz

6.1.7.11.2.2 File

class FileCls

File commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trace.iq.file.clone()
```

Subgroups

6.1.7.11.2.3 Repetition

class RepetitionCls

Repetition commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trace.iq.file.repetition.clone()
```

Subgroups

6.1.7.11.2.4 Count

SCPI Commands

```
TRACe:IQ:FILE:REPetition:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNT
value: float = driver.applications.k70Vsa.trace.iq.file.repetition.count.get()
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

return
repetition_count: integer

set(repetition_count: float) → None

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNT
driver.applications.k70Vsa.trace.iq.file.repetition.count.set(repetition_count_
↪= 1.0)
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

param repetition_count
integer

6.1.7.11.2.5 Wband

class WbandCls

Wband commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trace.iq.wband.clone()
```

Subgroups

6.1.7.11.2.6 Mbwidth

SCPI Commands

```
TRACe:IQ:WBAND:MBWidth
```

class MbwidthCls

Mbwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:WBAND:MBWidth
value: float = driver.applications.k70Vsa.trace.iq.wband.mbwidth.get()
```

Defines the maximum analysis bandwidth. Any value can be specified; the next higher fixed bandwidth is used.

return
max_bandwidth: No help available

set(max_bandwidth: float) → None

```
# SCPI: TRACe:IQ:WBAND:MBWidth
driver.applications.k70Vsa.trace.iq.wband.mbwidth.set(max_bandwidth = 1.0)
```

Defines the maximum analysis bandwidth. Any value can be specified; the next higher fixed bandwidth is used.

param max_bandwidth

80 MHz Restricts the analysis bandwidth to a maximum of 80 MHz. The bandwidth extension option R&S FSWP-B320 is deactivated. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to OFF. 160 MHz Restricts the analysis bandwidth to a maximum of 160 MHz. The bandwidth extension option R&S FSWP-B320 is deactivated. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to ON. 160 MHz | MAX

The bandwidth extension option is activated. The currently available maximum bandwidth is allowed. method RsFswp.Applications.IqAnalyzer.Trace.Iq.Wband.State.set is set to ON. Unit: Hz

6.1.7.11.2.7 State

SCPI Commands

TRACe:IQ:WBAND:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: TRACe:IQ:WBAND[:STATE]
value: bool = driver.applications.k70Vsa.trace.iq.wband.state.get()
```

This command determines whether the wideband provided by bandwidth extension options is used or not (if installed) .

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: TRACe:IQ:WBAND[:STATE]
driver.applications.k70Vsa.trace.iq.wband.state.set(state = False)
```

This command determines whether the wideband provided by bandwidth extension options is used or not (if installed) .

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.1.7.12 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.applications.k70Vsa.trigger.repcap_triggerPort_get()
driver.applications.k70Vsa.trigger.repcap_triggerPort_set(repcap.TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 12 total commands, 1 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.clone()
```

Subgroups

6.1.7.12.1 Sequence

class SequenceCls

Sequence commands group definition. 12 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.clone()
```

Subgroups

6.1.7.12.1.1 Dtime

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:DTIME
```

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:DTIME
value: float = driver.applications.k70Vsa.trigger.sequence.dtime.
↳ get(triggerPort = repcap.TriggerPort.Default)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

dropout_time: Dropout time of the trigger. Range: 0 s to 10.0 s, Unit: S

set(dropout_time: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:DTIME
driver.applications.k70Vsa.trigger.sequence.dtime.set(dropout_time = 1.0,
↳ triggerPort = repcap.TriggerPort.Default)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param dropout_time

Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.2 Holdoff**class HoldoffCls**

Holdoff commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.holdoff.clone()
```

Subgroups**6.1.7.12.1.3 Time****SCPI Commands**

```
TRIGger<TriggerPort>:SEquence:HOLDoff:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:HOLDoff[:TIME]
value: float = driver.applications.k70Vsa.trigger.sequence.holdoff.time.
↳get(triggerPort = repcap.TriggerPort.Default)
```

Defines the time offset between the trigger event and the start of the measurement.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

offset: The allowed range is 0 s to 30 s. Unit: S

set(offset: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:HOLDoff[:TIME]
driver.applications.k70Vsa.trigger.sequence.holdoff.time.set(offset = 1.0,
↳triggerPort = repcap.TriggerPort.Default)
```

Defines the time offset between the trigger event and the start of the measurement.

param offset

The allowed range is 0 s to 30 s. Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.4 IfPower**class IfPowerCls**

IfPower commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.ifPower.clone()
```

Subgroups**6.1.7.12.1.5 Holdoff****SCPI Commands**

```
TRIGger<TriggerPort>:SEquence:IFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:IFPower:HOLDoff
value: float = driver.applications.k70Vsa.trigger.sequence.ifPower.holdoff.
↪get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) . Note: If you perform gated measurements in combination with the IF Power trigger, the R&S FSWP ignores the holding time for frequency sweep, FFT sweep, zero span and I/Q data measurements.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

period: Range: 0 s to 10 s, Unit: S

set(period: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:IFPower:HOLDoff
driver.applications.k70Vsa.trigger.sequence.ifPower.holdoff.set(period = 1.0, ↪
↪triggerPort = repcap.TriggerPort.Default)
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) . Note: If you perform gated

measurements in combination with the IF Power trigger, the R&S FSWP ignores the holding time for frequency sweep, FFT sweep, zero span and I/Q data measurements.

param period

Range: 0 s to 10 s, Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.6 Hysteresis

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:IFPower:HYSTeresis
```

class HysteresisCls

Hysteresis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:IFPower:HYSTeresis
value: float = driver.applications.k70Vsa.trigger.sequence.ifPower.hysteresis.
↳get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

hysteresis: Range: 3 dB to 50 dB, Unit: DB

set(hysteresis: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:IFPower:HYSTeresis
driver.applications.k70Vsa.trigger.sequence.ifPower.hysteresis.set(hysteresis =
↳1.0, triggerPort = repcap.TriggerPort.Default)
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

param hysteresis

Range: 3 dB to 50 dB, Unit: DB

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.7 Level

class LevelCls

Level commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.level.clone()
```

Subgroups

6.1.7.12.1.8 External<ExternalPort>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.applications.k70Vsa.trigger.sequence.level.external.repcap_externalPort_get()
driver.applications.k70Vsa.trigger.sequence.level.external.repcap_externalPort_
↪set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:LEVel:EXTeRnal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(triggerPort=TriggerPort.Default, externalPort=ExternalPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel[:EXTeRnal<ap>]
value: float = driver.applications.k70Vsa.trigger.sequence.level.external.
↪get(triggerPort = repcap.TriggerPort.Default, externalPort = repcap.
↪ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

level_external: No help available

set(level_external: float, triggerPort=TriggerPort.Default, externalPort=ExternalPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel[:EXternal<ap>]
driver.applications.k70Vsa.trigger.sequence.level.external.set(level_external = 1.0, triggerPort = repcap.TriggerPort.Default, externalPort = repcap.ExternalPort.Default)
```

This command defines the level the external signal must exceed to cause a trigger event.

param level_external

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'External')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.level.external.clone()
```

6.1.7.12.1.9 IfPower

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:LEVel:IFPower
```

class IfPowerCls

IfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:IFPower
value: float = driver.applications.k70Vsa.trigger.sequence.level.ifPower.get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

level_if_power: No help available

set(level_if_power: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:IFPower
driver.applications.k70Vsa.trigger.sequence.level.ifPower.set(level_if_power =
↪1.0, triggerPort = repcap.TriggerPort.Default)
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param level_if_power

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.12.1.10 IqPower

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:LEVel:IQPower
```

class IqPowerCls

IqPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:IQPower
value: float = driver.applications.k70Vsa.trigger.sequence.level.iqPower.
↪get(triggerPort = repcap.TriggerPort.Default)
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

level_iq_power: No help available

set(level_iq_power: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:IQPower
driver.applications.k70Vsa.trigger.sequence.level.iqPower.set(level_iq_power =
↪1.0, triggerPort = repcap.TriggerPort.Default)
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param level_iq_power

Range: -130 dBm to 30 dBm, Unit: DBM

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.11 RfPower**SCPI Commands**

```
TRIGger<TriggerPort>:SEquence:LEVel:RFPower
```

class RfPowerCls

RfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:RFPower
value: float = driver.applications.k70Vsa.trigger.sequence.level.rfPower.
↳get(triggerPort = repcap.TriggerPort.Default)
```

No command help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

level_rf_power: No help available

set(level_rf_power: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEVel:RFPower
driver.applications.k70Vsa.trigger.sequence.level.rfPower.set(level_rf_power =
↳1.0, triggerPort = repcap.TriggerPort.Default)
```

No command help available

param level_rf_power

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.1.7.12.1.12 Video**SCPI Commands**

```
TRIGger<TriggerPort>:SEquence:LEVel:VIDEO
```

class VideoCls

Video commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:LEvel:VIDeo
value: float = driver.applications.k70Vsa.trigger.sequence.level.video.
↪get(triggerPort = repcap.TriggerPort.Default)
```

No command help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

arg_0: No help available

set(arg_0: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:LEvel:VIDeo
driver.applications.k70Vsa.trigger.sequence.level.video.set(arg_0 = 1.0, ↪
↪triggerPort = repcap.TriggerPort.Default)
```

No command help available

param arg_0

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.12.1.13 RfPower

class RfPowerCls

RfPower commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.rfPower.clone()
```

Subgroups

6.1.7.12.1.14 Holdoff

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:RFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:RFPower:HOLDoff
value: float = driver.applications.k70Vsa.trigger.sequence.rfPower.holdoff.
↳get(triggerPort = repcap.TriggerPort.Default)
```

No command help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

time: No help available

set(time: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:RFPower:HOLDoff
driver.applications.k70Vsa.trigger.sequence.rfPower.holdoff.set(time = 1.0,↳
↳triggerPort = repcap.TriggerPort.Default)
```

No command help available

param time

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.12.1.15 Slope

SCPI Commands

TRIGger<TriggerPort>:SEquence:SLOPe

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → SlopeType

```
# SCPI: TRIGger<tp>[:SEquence]:SLOPe
value: enums.SlopeType = driver.applications.k70Vsa.trigger.sequence.slope.
↳get(triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger slope.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

type_py: POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

set(type_py: SlopeType, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:SLOPe
driver.applications.k70Vsa.trigger.sequence.slope.set(type_py = enums.SlopeType.
↳NEGative, triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger slope.

param type_py

POSitive|NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.1.7.12.1.16 Time

class TimeCls

Time commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.applications.k70Vsa.trigger.sequence.time.clone()
```

Subgroups

6.1.7.12.1.17 Rinterval

SCPI Commands

```
TRIGger<TriggerPort>:SEquence:TIME:RINTerval
```

class RintervalCls

Rinterval commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: TRIGger<tp>[:SEquence]:TIME:RINTerval
value: float = driver.applications.k70Vsa.trigger.sequence.time.rinterval.
↳get(triggerPort = repcap.TriggerPort.Default)
```

No command help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

interval: No help available

set(interval: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: TRIGger<tp>[:SEquence]:TIME:RINterval
driver.applications.k70Vsa.trigger.sequence.time.rinterval.set(interval = 1.0,
↪ triggerPort = repcap.TriggerPort.Default)
```

No command help available

param interval

No help available

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.2 Calculate<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.calculate.repcap_window_get()
driver.calculate.repcap_window_set(repcap.Window.Nr1)
```

class CalculateCls

Calculate commands group definition. 298 total commands, 13 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.clone()
```

Subgroups

6.2.1 DeltaMarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.calculate.deltaMarker.repcap_deltaMarker_get()
driver.calculate.deltaMarker.repcap_deltaMarker_set(repcap.DeltaMarker.Nr1)
```

class DeltaMarkerCls

DeltaMarker commands group definition. 44 total commands, 13 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.clone()
```

Subgroups

6.2.1.1 Aoff

SCPI Commands

```
CALCulate<Window>:DELTaMarker:AOff
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker:AOff
driver.calculate.deltaMarker.aoff.set(window = repcap.Window.Default)
```

This command turns off all delta markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.2 Function

class FunctionCls

Function commands group definition. 14 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.clone()
```

Subgroups

6.2.1.2.1 AfPhase

class AfPhaseCls

AfPhase commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.afPhase.clone()
```

Subgroups

6.2.1.2.1.1 Result

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:AFPHase:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:AFPHase:RESult
value: float = driver.calculate.deltaMarker.function.afPhase.result.get(window_
↳ repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

af_phase: No help available

6.2.1.2.1.2 State

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:AFPHase:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:AFPHase[:STATe]
value: bool = driver.calculate.deltaMarker.function.afPhase.state.get(window =
↳ repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

state: No help available

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNction:AFPHase[:STATe]
driver.calculate.deltaMarker.function.afPhase.state.set(state = False, window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.2.1.2.2 Bpower

class BpowerCls

Bpower commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.bpower.clone()
```

Subgroups

6.2.1.2.2.1 Mode

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:FUNction:BPOWer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → MarkerMode

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNCTION:BPOWer:MODE
value: enums.MarkerMode = driver.calculate.deltaMarker.function.bpower.mode.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

mode: No help available

set(*mode: MarkerMode, window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNCTION:BPOWer:MODE
driver.calculate.deltaMarker.function.bpower.mode.set(mode = enums.MarkerMode.
↪DENSity, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↪Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.2.1.2.2.2 Result

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNCTION:BPOWer:RESult
value: float = driver.calculate.deltaMarker.function.bpower.result.get(window =
↪repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

power: No help available

6.2.1.2.2.3 Span

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:SPAN`

class SpanCls

Span commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=*Window.Default*, *deltaMarker*=*DeltaMarker.Default*) → float

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNCTION:BPOWer:SPAN
value: float = driver.calculate.deltaMarker.function.bpower.span.get(window = ↵
↵repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

span: No help available

set(*span*: float, *window*=*Window.Default*, *deltaMarker*=*DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:FUNCTION:BPOWer:SPAN
driver.calculate.deltaMarker.function.bpower.span.set(span = 1.0, window = ↵
↵repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param span

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.2.2.4 State

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:BPOWer:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:BPOWer[:STATe]
value: bool = driver.calculate.deltaMarker.function.bpower.state.get(window = ↪
↪repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

state: No help available

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:BPOWer[:STATe]
driver.calculate.deltaMarker.function.bpower.state.set(state = False, window = ↪
↪repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.2.3 Fixed

class FixedCls

Fixed commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.fixed.clone()
```

Subgroups

6.2.1.2.3.1 Rpoint

class RpointCls

Rpoint commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.fixed.rpoint.clone()
```

Subgroups

6.2.1.2.3.2 Maximum

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.fixed.rpoint.maximum.clone()
```

Subgroups

6.2.1.2.3.3 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:FIXed:RPOint:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed:RPOint:MAXimum[:PEAK]
driver.calculate.deltaMarker.function.fixed.rpoint.maximum.peak.set(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.2.1.2.3.4 X

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:FIXed:RPOint:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed:RPOint:X
value: float = driver.calculate.deltaMarker.function.fixed.rpoint.x.get(window,
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

ref_point: No help available

set(ref_point: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed:RPOint:X
driver.calculate.deltaMarker.function.fixed.rpoint.x.set(ref_point = 1.0,
↳window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param ref_point

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.2.1.2.3.5 Y**SCPI Commands**

CALCulate<Window>:DELTamarker<DeltaMarker>:FUNction:FIXed:RPOint:Y
--

class YCls

Y commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

<pre># SCPI: CALCulate<n>:DELTamarker<m>:FUNction:FIXed:RPOint:Y value: float = driver.calculate.deltaMarker.function.fixed.rpoint.y.get(window, ↪ = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

ref_point: No help available

set(ref_point: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

<pre># SCPI: CALCulate<n>:DELTamarker<m>:FUNction:FIXed:RPOint:Y driver.calculate.deltaMarker.function.fixed.rpoint.y.set(ref_point = 1.0, ↪ window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)</pre>
--

No command help available

param ref_point

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.fixed.rpoint.y.clone()
```

Subgroups

6.2.1.2.3.6 Offset

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:FIXed:RPoInt:Y:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed:RPoInt:Y:OFFSet
value: float = driver.calculate.deltaMarker.function.fixed.rpoint.y.offset.
↳get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

yvalue_relative: No help available

set(yvalue_relative: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed:RPoInt:Y:OFFSet
driver.calculate.deltaMarker.function.fixed.rpoint.y.offset.set(yvalue_relative,
↳= 1.0, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.
↳Default)
```

No command help available

param yvalue_relative

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.2.3.7 State

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:FIXed:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=*Window.Default*, *deltaMarker*=*DeltaMarker.Default*) → bool

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed[:STATe]
value: bool = driver.calculate.deltaMarker.function.fixed.state.get(window = ↵
↵repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

state: No help available

set(*state*: bool, *window*=*Window.Default*, *deltaMarker*=*DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:FIXed[:STATe]
driver.calculate.deltaMarker.function.fixed.state.set(state = False, window = ↵
↵repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.2.1.2.4 Pnoise

class PnoiseCls

Pnoise commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.function.pnoise.clone()
```

Subgroups

6.2.1.2.4.1 Auto

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:PNOise:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:PNOise:AUTO
driver.calculate.deltaMarker.function.pnoise.auto.set(state = False, window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.2.4.2 Result

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:PNOise:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:PNOise:RESult
value: float = driver.calculate.deltaMarker.function.pnoise.result.get(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

phasenoise: No help available

6.2.1.2.4.3 State

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:FUNction:PNOise:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → bool

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:PNOise[:STATe]
value: bool = driver.calculate.deltaMarker.function.pnoise.state.get(window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

state: No help available

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:FUNction:PNOise[:STATe]
driver.calculate.deltaMarker.function.pnoise.state.set(state = False, window =
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.3 LinkTo**class LinkToCls**

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.linkTo.clone()
```

Subgroups**6.2.1.3.1 Marker<MarkerDestination>****RepCap Settings**

```
# Range: Nr1 .. Nr16
rc = driver.calculate.deltaMarker.linkTo.marker.repcap_markerDestination_get()
driver.calculate.deltaMarker.linkTo.marker.repcap_markerDestination_set(repcap.
↳MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:LINK:TO:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

set(state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<ms>:LINK:TO:MARKer<mt>
driver.calculate.deltaMarker.linkTo.marker.set(state = False, window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default, markerDestination =
↳repcap.MarkerDestination.Default)
```

This command links the delta source marker <ms> to any active destination marker <md> (normal or delta marker).

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.linkTo.marker.clone()
```

6.2.1.4 Maximum**class MaximumCls**

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.maximum.clone()
```

Subgroups**6.2.1.4.1 Left****SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:LEFT
driver.calculate.deltaMarker.maximum.left.set(window = repcap.Window.Default,
↵ deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.4.2 Next**SCPI Commands**

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum:NEXT
driver.calculate.deltaMarker.maximum.next.set(window = repcap.Window.Default,
deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next positive peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.4.3 Peak**SCPI Commands**

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MAXimum[:PEAK]
driver.calculate.deltaMarker.maximum.peak.set(window = repcap.Window.Default,
deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.4.4 Right

SCPI Commands

`CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MAXimum:RIGHT
driver.calculate.deltaMarker.maximum.right.set(window = repcap.Window.Default,
deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next positive peak value on the trace. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.5 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.minimum.clone()
```

Subgroups

6.2.1.5.1 Left

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:LEFT
driver.calculate.deltaMarker.minimum.left.set(window = repcap.Window.Default,
↳ deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.5.2 Next

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MINimum:NEXT
driver.calculate.deltaMarker.minimum.next.set(window = repcap.Window.Default,
↳ deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.5.3 Peak**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MINimum[:PEAK]
driver.calculate.deltaMarker.minimum.peak.set(window = repcap.Window.Default,
↳ deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.5.4 Right**SCPI Commands**

```
CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MINimum:RIGHT
driver.calculate.deltaMarker.minimum.right.set(window = repcap.Window.Default,
↳ deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.1.6 Mode

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → ReferenceMode

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MODE
value: enums.ReferenceMode = driver.calculate.deltaMarker.mode.get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command defines whether the position of a delta marker is provided as an absolute value or relative to a reference marker. Note that this setting applies to all windows. Note that when the position of a delta marker is queried, the result is always an absolute value (see method RsFswp.Applications.K30_NoiseFigure.Calculate.DeltaMarker.X.set) !

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

reference_mode: No help available

set(reference_mode: ReferenceMode, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MODE
driver.calculate.deltaMarker.mode.set(reference_mode = enums.ReferenceMode.ABSolute, window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command defines whether the position of a delta marker is provided as an absolute value or relative to a reference marker. Note that this setting applies to all windows. Note that when the position of a delta marker is queried, the result is always an absolute value (see method RsFswp.Applications.K30_NoiseFigure.Calculate.DeltaMarker.X.set) !

param reference_mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.7 Mref

SCPI Commands

CALCulate<Window>:DELTamarker<DeltaMarker>:MREF

class MrefCls

Mref commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → int

```
# SCPI: CALCulate<n>:DELTamarker<m>:MREF
value: int = driver.calculate.deltaMarker.mref.get(window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

marker_name: No help available

set(marker_name: int, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:MREF
driver.calculate.deltaMarker.mref.set(marker_name = 1, window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1.

param marker_name

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.8 Mreference

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:MREference

class MreferenceCls

Mreference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MREference
value: float = driver.calculate.deltaMarker.mreference.get(window = repcap.
↳ Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

reference: No help available

set(reference: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:MREference
driver.calculate.deltaMarker.mreference.set(reference = 1.0, window = repcap.
↳ Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects a reference marker for a delta marker other than marker 1.

param reference

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.9 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.clone()
```

Subgroups

6.2.1.9.1 Frame

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:FRAME
```

class FrameCls

Frame commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:FRAME
value: float = driver.calculate.deltaMarker.spectrogram.frame.get(window = Window
↳repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

frame: No help available

set(frame: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:FRAME
driver.calculate.deltaMarker.spectrogram.frame.set(frame = 1.0, window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param frame

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.9.2 Sarea

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:SARea

class SareaCls

Sarea commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → SearchArea

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:SARea
value: enums.SearchArea = driver.calculate.deltaMarker.spectrogram.sarea.
↪get(window = repcap.Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

search_area: No help available

set(*search_area: SearchArea, window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:SARea
driver.calculate.deltaMarker.spectrogram.sarea.set(search_area = enums.
↪SearchArea.MEMory, window = repcap.Window.Default, deltaMarker = repcap.
↪DeltaMarker.Default)
```

No command help available

param search_area

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.9.3 Xy

class XyCls

Xy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.xy.clone()
```

Subgroups

6.2.1.9.3.1 Maximum

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.xy.maximum.clone()
```

Subgroups

6.2.1.9.3.2 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:XY:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:XY:MAXimum[:PEAK]
driver.calculate.deltaMarker.spectrogram.xy.maximum.peak.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

```
set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) →
None
```

6.2.1.9.3.3 Minimum

class MinimumCls

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.xy.minimum.clone()
```

Subgroups

6.2.1.9.3.4 Peak

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:XY:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None
```

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:XY:MINimum[:PEAK]
driver.calculate.deltaMarker.spectrogram.xy.minimum.peak.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

```
set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) →
None
```

6.2.1.9.4 Y

class YCls

Y commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.y.clone()
```

Subgroups

6.2.1.9.4.1 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.y.maximum.clone()
```

Subgroups

6.2.1.9.4.2 Above

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtrogram:Y:MAXimum:ABOVE
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:SPECtrogram:Y:MAXimum:ABOVE
driver.calculate.deltaMarker.spectrogram.y.maximum.above.set(window = repcap.
↪Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.3 Below

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum:BELOW
driver.calculate.deltaMarker.spectrogram.y.maximum.below.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.4 Next

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MAXimum:NEXT
driver.calculate.deltaMarker.spectrogram.y.maximum.next.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.5 Peak**SCPI Commands**

`CALCulate<Window>:DELTamarker<DeltaMarker>:SPECTrogram:Y:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECTrogram:Y:MAXimum[:PEAK]
driver.calculate.deltaMarker.spectrogram.y.maximum.peak.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.6 Minimum**class MinimumCls**

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.spectrogram.y.minimum.clone()
```

Subgroups

6.2.1.9.4.7 Above

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MINimum:ABOve
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MINimum:ABOve
driver.calculate.deltaMarker.spectrogram.y.minimum.above.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1) → None

6.2.1.9.4.8 Below

SCPI Commands

```
CALCulate<Window>:DELTamarker<DeltaMarker>:SPECtrogram:Y:MINimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>:SPECtrogram:Y:MINimum:BELOW
driver.calculate.deltaMarker.spectrogram.y.minimum.below.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.9 Next

SCPI Commands

CALCulate<Window>:DELTAmarker<DeltaMarker>:SPECtrogram:Y:MINimum:NEXT

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>:SPECtrogram:Y:MINimum:NEXT
driver.calculate.deltaMarker.spectrogram.y.minimum.next.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.9.4.10 Peak

SCPI Commands

CALCulate<Window>:DELTAmarker<DeltaMarker>:SPECtrogram:Y:MINimum:PEAK

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>:SPECtrogram:Y:MINimum[:PEAK]
driver.calculate.deltaMarker.spectrogram.y.minimum.peak.set(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

set_with_opc(*window=Window.Default, deltaMarker=DeltaMarker.Default, opc_timeout_ms: int = -1*) → None

6.2.1.10 State**SCPI Commands**

CALCulate<Window>:DELTamarker<DeltaMarker>:STATe
--

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → bool

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
value: bool = driver.calculate.deltaMarker.state.get(window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTamarker<m>[:STATe]
driver.calculate.deltaMarker.state.set(state = False, window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command turns delta markers on and off. If necessary, the command activates the delta marker first. No suffix at DELTmarker turns on delta marker 1.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.1.11 Trace

SCPI Commands

`CALCulate<Window>:DELTaMarker<DeltaMarker>:TRACe`

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
value: float = driver.calculate.deltaMarker.trace.get(window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than ‘Blank’. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

trace: Trace number the marker is assigned to.

set(*trace: float, window=Window.Default, deltaMarker=DeltaMarker.Default*) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:TRACe
driver.calculate.deltaMarker.trace.set(trace = 1.0, window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command selects the trace a delta marker is positioned on. Note that the corresponding trace must have a trace mode other than ‘Blank’. If necessary, the command activates the marker first.

param trace

Trace number the marker is assigned to.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

6.2.1.12 X

SCPI Commands

CALCulate<Window>:DELTaMarker<DeltaMarker>:X

class XCls

X commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
value: float = driver.calculate.deltaMarker.x.get(window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

return

stimulus: No help available

set(stimulus: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X
driver.calculate.deltaMarker.x.set(stimulus = 1.0, window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command moves a delta marker to a particular coordinate on the x-axis. If necessary, the command activates the delta marker and positions a reference marker to the peak power.

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘DeltaMarker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.deltaMarker.x.clone()
```

Subgroups

6.2.1.12.1 Relative

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:X:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:X:RELative
value: float = driver.calculate.deltaMarker.x.relative.get(window = repcap.
↳Window.Default, deltaMarker = repcap.DeltaMarker.Default)
```

This command queries the relative position of a delta marker on the x-axis. If necessary, the command activates the delta marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

xvalue_relative: No help available

6.2.1.13 Y

SCPI Commands

```
CALCulate<Window>:DELTaMarker<DeltaMarker>:Y
```

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, deltaMarker=DeltaMarker.Default) → float

```
# SCPI: CALCulate<n>:DELTaMarker<m>:Y
value: float = driver.calculate.deltaMarker.y.get(window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

return

position: No help available

set(position: float, window=Window.Default, deltaMarker=DeltaMarker.Default) → None

```
# SCPI: CALCulate<n>:DELTAmarker<m>:Y
driver.calculate.deltaMarker.y.set(position = 1.0, window = repcap.Window.
↳Default, deltaMarker = repcap.DeltaMarker.Default)
```

Queries the result at the position of the specified delta marker.

param position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param deltaMarker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'DeltaMarker')

6.2.2 Spectrum

class SpectrumCls

Spectrum commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrum.clone()
```

Subgroups

6.2.2.1 PeakSearch

class PeakSearchCls

PeakSearch commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.espectrum.peakSearch.clone()
```

Subgroups

6.2.2.1.1 Auto

SCPI Commands

```
CALCulate<Window>:ESpectrum:PSEarch:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch:AUTO
driver.calculate.espectrum.peakSearch.auto.set(state = False, window = repcap.
↳Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.2.1.2 Immediate

SCPI Commands

```
CALCulate<Window>:ESpectrum:PSEarch:IMMEDIATE
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default) → None

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch[:IMMEDIATE]
driver.calculate.espectrum.peakSearch.immediate.set(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

set_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

6.2.2.1.3 Margin

SCPI Commands

```
CALCulate<Window>:ESpectrum:PSEarch:MARGin
```

class MarginCls

Margin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch:MARGin
value: float = driver.calculate.espectrum.peakSearch.margin.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

threshold: No help available

set(*threshold: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch:MARGin
driver.calculate.espectrum.peakSearch.margin.set(threshold = 1.0, window =
↳repcap.Window.Default)
```

No command help available

param threshold

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.2.1.4 Pshow

SCPI Commands

```
CALCulate<Window>:ESpectrum:PSEarch:PSHow
```

class PshowCls

Pshow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch:PSHow
value: bool = driver.calculate.espectrum.peakSearch.pshow.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:ESpectrum:PSEarch:PSHow
driver.calculate.espectrum.peakSearch.pshow.set(state = False, window = repcap.
↳Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.3 Limit<LimitIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.calculate.limit.repcap_limitIx_get()
driver.calculate.limit.repcap_limitIx_set(repcap.LimitIx.Nr1)
```

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:DELeTe
```

class LimitCls

Limit commands group definition. 79 total commands, 14 Subgroups, 1 group commands Repeated Capability: LimitIx, default value after init: LimitIx.Nr1

delete(window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:DELeTe
driver.calculate.limit.delete(window = repcap.Window.Default, limitIx = repcap.
↳LimitIx.Default)
```

This command deletes a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

delete_with_opc(window=Window.Default, limitIx=LimitIx.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.clone()
```

Subgroups

6.2.3.1 AcPower

class AcPowerCls

AcPower commands group definition. 37 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.clone()
```

Subgroups

6.2.3.1.1 Achannel

class AchannelCls

Achannel commands group definition. 7 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.achannel.clone()
```

Subgroups

6.2.3.1.1.1 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPpower:ACHannel:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class Limits

Response structure. Fields:

- Lower_Limit: float: No parameter help available
- Upper_Limit: float: No parameter help available

get(*window=Window.Default, limitIx=LimitIx.Default*) → Limits

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowe:rACHannel:ABSolute
value: Limits = driver.calculate.limit.acPower.achannel.absolute.get(window = ↵
↵repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

structure: for return value, see the help for Limits structure arguments.

set(*lower_limit: float, upper_limit: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowe:rACHannel:ABSolute
driver.calculate.limit.acPower.achannel.absolute.set(lower_limit = 1.0, upper_
↵limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param lower_limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.achannel.absolute.clone()
```

Subgroups

6.2.3.1.1.2 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowe:rACHannel:ABSolute:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class State

Response structure. Fields:

- State_Lower: bool: No parameter help available
- State_Upper: bool: No parameter help available

get(window=*Window.Default*, limitIx=*LimitIx.Default*) → State

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:ACHannel:ABSolute:STATe
value: State = driver.calculate.limit.acPower.achannel.absolute.state.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for State structure arguments.

set(state_lower: bool, state_upper: Optional[bool] = None, window=*Window.Default*, limitIx=*LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:ACHannel:ABSolute:STATe
driver.calculate.limit.acPower.achannel.absolute.state.set(state_lower = False, ↪
↪state_upper = False, window = repcap.Window.Default, limitIx = repcap.LimitIx.
↪Default)
```

No command help available

param state_lower

No help available

param state_upper

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.1.1.3 Relative

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel:RELative

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class RelativeStruct

Response structure. Fields:

- Lower_Limit: float: No parameter help available
- Upper_Limit: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default) → RelativeStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:ACHannel[:RELative]
value: RelativeStruct = driver.calculate.limit.acPower.achannel.relative.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for RelativeStruct structure arguments.

set(lower_limit: float, upper_limit: Optional[float] = None, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:ACHannel[:RELative]
driver.calculate.limit.acPower.achannel.relative.set(lower_limit = 1.0, upper_
↪limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param lower_limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.achannel.relative.clone()
```

Subgroups

6.2.3.1.1.4 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class State

Response structure. Fields:

- State_Lower: bool: No parameter help available
- State_Upper: bool: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default) → State

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:ACHannel[:RELative]:STATe
value: State = driver.calculate.limit.acPower.achannel.relative.state.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for State structure arguments.

set(state_lower: bool, state_upper: Optional[bool] = None, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:ACHannel[:RELative]:STATe
driver.calculate.limit.acPower.achannel.relative.state.set(state_lower = False,
↪state_upper = False, window = repcap.Window.Default, limitIx = repcap.LimitIx.
↪Default)
```

No command help available

param state_lower

No help available

param state_upper

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.1.1.5 Result

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:ACPoweR:ACHannel:RESult`

class ResultCls

Result commands group definition. 3 total commands, 2 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=*Window.Default*, limitIx=*LimitIx.Default*) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ACHannel:RESult
value: GetStruct = driver.calculate.limit.acPower.achannel.result.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for GetStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.achannel.result.clone()
```

Subgroups

6.2.3.1.1.6 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowe:rACHannel:RESult:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowe:rACHannel:RESult:ABSolute
value: GetStruct = driver.calculate.limit.acPower.achannel.result.absolute.
get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.1.7 Relative

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowe:rACHannel:RESult:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:ACHannel:RESult:RELative
value: GetStruct = driver.calculate.limit.acPower.achannel.result.relative.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.2 Alternate<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.calculate.limit.acPower.alternate.repcap_channel_get()
driver.calculate.limit.acPower.alternate.repcap_channel_set(repcap.Channel.Ch1)
```

class AlternateCls

Alternate commands group definition. 7 total commands, 3 Subgroups, 0 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.alternate.clone()
```

Subgroups

6.2.3.1.2.1 Absolute

SCPI Commands

```
CALCulate<Window>:LIMIT<LimitIx>:ACPower:ALternate<Channel>:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class AbsoluteStruct

Response structure. Fields:

- Lower_Limit: float: No parameter help available
- Upper_Limit: float: No parameter help available

get(*window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default*) → AbsoluteStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>:ABSolute
value: AbsoluteStruct = driver.calculate.limit.acPower.alternate.absolute.
↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪ = repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface ‘Alternate’)

return

structure: for return value, see the help for AbsoluteStruct structure arguments.

set(*lower_limit: float, upper_limit: Optional[float] = None, window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>:ABSolute
driver.calculate.limit.acPower.alternate.absolute.set(lower_limit = 1.0, upper_
↪ limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↪ channel = repcap.Channel.Default)
```

No command help available

param lower_limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface ‘Alternate’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.alternate.absolute.clone()
```

Subgroups

6.2.3.1.2.2 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:ALternate<Channel>:ABSolute:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class State

Response structure. Fields:

- State_Lower: bool: No parameter help available
- State_Upper: bool: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → State

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>:ABSolute:STATe
value: State = driver.calculate.limit.acPower.alternate.absolute.state.
↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪ = repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface ‘Alternate’)

return

structure: for return value, see the help for State structure arguments.

set(state_lower: bool, state_upper: Optional[bool] = None, window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>:ABSolute:STATe
driver.calculate.limit.acPower.alternate.absolute.state.set(state_lower = False,
↪ state_upper = False, window = repcap.Window.Default, limitIx = repcap.
↪ LimitIx.Default, channel = repcap.Channel.Default)
```

No command help available

param state_lower

No help available

param state_upper

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

6.2.3.1.2.3 Relative**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:ACPoweR:ALternate<Channel>:RELative

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class RelativeStruct

Response structure. Fields:

- Lower_Limit: float: No parameter help available
- Upper_Limit: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → RelativeStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>[:RELative]
value: RelativeStruct = driver.calculate.limit.acPower.alternate.relative.
↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪ = repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return

structure: for return value, see the help for RelativeStruct structure arguments.

```
set(lower_limit: float, upper_limit: Optional[float] = None, window=Window.Default,
    limitIx=LimitIx.Default, channel=Channel.Default) → None
```

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:ALternate<ch>[:RELative]
driver.calculate.limit.acPower.alternate.relative.set(lower_limit = 1.0, upper_
↪ limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↪ channel = repcap.Channel.Default)
```

No command help available

param lower_limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface ‘Alternate’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.alternate.relative.clone()
```

Subgroups

6.2.3.1.2.4 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowEr:ALternate<Channel>:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class State

Response structure. Fields:

- State_Lower: bool: No parameter help available
- State_Upper: bool: No parameter help available

```
get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → State
```

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>[:RELative]:STATe
value: State = driver.calculate.limit.acPower.alternate.relative.state.
↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪ = repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return

structure: for return value, see the help for State structure arguments.

set(state_lower: bool, state_upper: Optional[bool] = None, window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>[:RELative]:STATe
driver.calculate.limit.acPower.alternate.relative.state.set(state_lower = False,
↪ state_upper = False, window = repcap.Window.Default, limitIx = repcap.
↪ LimitIx.Default, channel = repcap.Channel.Default)
```

No command help available

param state_lower

No help available

param state_upper

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

6.2.3.1.2.5 Result

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:ACPoweR:ALternate<Channel>:RESult`

class ResultCls

Result commands group definition. 3 total commands, 2 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:ALternate<ch>:RESult
value: GetStruct = driver.calculate.limit.acPower.alternate.result.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel = repcap.
↳Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface ‘Alternate’)

return

structure: for return value, see the help for GetStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.alternate.result.clone()
```

Subgroups

6.2.3.1.2.6 Absolute

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:ACPoweR:ALternate<Channel>:RESult:ABSolute`

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:ALternate<ch>:RESult:ABSolute
value: GetStruct = driver.calculate.limit.acPower.alternate.result.absolute.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪= repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.2.7 Relative**SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:ACPpower:ALternate<Channel>:RESult:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, channel=Channel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:ALternate<ch>:RESult:RELative
value: GetStruct = driver.calculate.limit.acPower.alternate.result.relative.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, channel_
↪= repcap.Channel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.3 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.calculate.limit.acPower.gap.repcap_gapChannel_get()
driver.calculate.limit.acPower.gap.repcap_gapChannel_set(repcap.GapChannel.Nr1)
```

class GapCls

Gap commands group definition. 22 total commands, 4 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.clone()
```

Subgroups

6.2.3.1.3.1 Aclr

class AclrCls

Aclr commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.aclr.clone()
```


Subgroups

6.2.3.1.3.2 Result

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:ACLR:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Upper_Limit: enums.CheckResult: No parameter help available
- Lower_Limit: enums.CheckResult: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:ACLR:RESult
value: GetStruct = driver.calculate.limit.acPower.gap.aclr.result.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel = repcap.
↳GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.3.3 Auto

class AutoCls

Auto commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.clone()
```

Subgroups

6.2.3.1.3.4 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:AUTO:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class AbsoluteStruct

Response structure. Fields:

- Limit: float: No parameter help available
- Reserved: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → AbsoluteStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO]:ABSolute
value: AbsoluteStruct = driver.calculate.limit.acPower.gap.auto.absolute.
↳get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↳gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for AbsoluteStruct structure arguments.

set(limit: float, reserved: Optional[float] = None, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO]:ABSolute
driver.calculate.limit.acPower.gap.auto.absolute.set(limit = 1.0, reserved = 1.
↳0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↳gapChannel = repcap.GapChannel.Default)
```

No command help available

param limit

No help available

param reserved

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.absolute.clone()
```

Subgroups

6.2.3.1.3.5 State

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:AUTO:ABSolute:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO]:ABSolute:STATe
value: bool = driver.calculate.limit.acPower.gap.auto.absolute.state.get(window_
↪= repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel = ↪
↪repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

state: No help available

set(state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:GAP<gap>[:AUTO]:ABSolute:STATE
driver.calculate.limit.acPower.gap.auto.absolute.state.set(state = False,
↳ window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel
↳ = repcap.GapChannel.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

6.2.3.1.3.6 Aclr

class AclrCls

Aclr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.aclr.clone()
```

Subgroups

6.2.3.1.3.7 Relative

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPpower:GAP<GapChannel>:AUTO:ACLR:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class RelativeStruct

Response structure. Fields:

- Limit: float: No parameter help available

- Upper_Limit: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → RelativeStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:GAP<gap>[:AUTO]:ACLR[:RELative]
value: RelativeStruct = driver.calculate.limit.acPower.gap.auto.acLr.relative.
↪ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, ↪
↪ gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

structure: for return value, see the help for RelativeStruct structure arguments.

set(limit: float, upper_limit: Optional[float] = None, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:GAP<gap>[:AUTO]:ACLR[:RELative]
driver.calculate.limit.acPower.gap.auto.acLr.relative.set(limit = 1.0, upper_
↪ limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↪ gapChannel = repcap.GapChannel.Default)
```

No command help available

param limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.aclr.relative.clone()
```

Subgroups

6.2.3.1.3.8 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:AUTO:ACLR:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO]:ACLR[:RELative]:STATe
value: bool = driver.calculate.limit.acPower.gap.auto.aclr.relative.state.
↳ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↳ gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

state: No help available

set(state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO]:ACLR[:RELative]:STATe
driver.calculate.limit.acPower.gap.auto.aclr.relative.state.set(state = False,
↳ window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel
↳ = repcap.GapChannel.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.2.3.1.3.9 Caclr**class CaclrCls**

Caclr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.caclr.clone()
```

Subgroups**6.2.3.1.3.10 Relative****SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:AUTO:CACLr:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class RelativeStruct

Response structure. Fields:

- Limit: float: No parameter help available
- Upper_Limit: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → RelativeStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:AUTO][:CACLr][:RELative]
value: RelativeStruct = driver.calculate.limit.acPower.gap.auto.caclr.relative.
↳ get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↳ gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for RelativeStruct structure arguments.

set(*limit: float, upper_limit: Optional[float] = None, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:GAP<gap>[:AUTO][:CACLR][:RELative]
driver.calculate.limit.acPower.gap.auto.cacLR.relative.set(limit = 1.0, upper_
↪ limit = 1.0, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↪ gapChannel = repcap.GapChannel.Default)
```

No command help available

param limit

No help available

param upper_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.auto.cacLR.relative.clone()
```

Subgroups

6.2.3.1.3.11 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:AUTO:CACLR:RELative:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default*) → bool


```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>[:AUTO][:CAClr][:RELative]:STATE
value: bool = driver.calculate.limit.acPower.gap.auto.caclr.relative.state.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, ↪
↪gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

state: No help available

set(state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>[:AUTO][:CAClr][:RELative]:STATE
driver.calculate.limit.acPower.gap.auto.caclr.relative.state.set(state = False, ↪
↪window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel ↪
↪= repcap.GapChannel.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

6.2.3.1.3.12 Caclr

class CaclrCls

Caclr commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.caclr.clone()
```

Subgroups

6.2.3.1.3.13 Result

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:CACLR:RESult
```

class ResultCls

Result commands group definition. 3 total commands, 2 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Lower_Gap_Ab: float: No parameter help available
- Upper_Gap_Ab: float: No parameter help available
- Lower_Gap_Bc: float: No parameter help available
- Upper_Gap_Bc: float: No parameter help available
- Lower_Gap_Cd: float: No parameter help available
- Upper_Gap_Cd: float: No parameter help available
- Lower_Gap_De: float: No parameter help available
- Upper_Gap_De: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:CACLR]:RESult
value: GetStruct = driver.calculate.limit.acPower.gap.caclr.result.get(window =
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default, gapChannel = repcap.
↪GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for GetStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.caclr.result.clone()
```

Subgroups

6.2.3.1.3.14 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:CACLR:RESult:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Lower_Gap_Ab: float: No parameter help available
- Upper_Gap_Ab: float: No parameter help available
- Lower_Gap_Bc: float: No parameter help available
- Upper_Gap_Bc: float: No parameter help available
- Lower_Gap_Cd: float: No parameter help available
- Upper_Gap_Cd: float: No parameter help available
- Lower_Gap_De: float: No parameter help available
- Upper_Gap_De: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:GAP<gap>[:CACLR]:RESult:ABSolute
value: GetStruct = driver.calculate.limit.acPower.gap.caclr.result.absolute.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default,
↪gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.3.15 Relative

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:CACLR:RESult:RELative
--

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Lower_Gap_Ab: float: No parameter help available
- Upper_Gap_Ab: float: No parameter help available
- Lower_Gap_Bc: float: No parameter help available
- Upper_Gap_Bc: float: No parameter help available
- Lower_Gap_Cd: float: No parameter help available
- Upper_Gap_Cd: float: No parameter help available
- Lower_Gap_De: float: No parameter help available
- Upper_Gap_De: float: No parameter help available

get(window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → GetStruct

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>[:CACLR]:RESult:RELative
value: GetStruct = driver.calculate.limit.acPower.gap.caclr.result.relative.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, ↪
↪gapChannel = repcap.GapChannel.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.3.1.3.16 Manual

class ManualCls

Manual commands group definition. 12 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.clone()
```

Subgroups

6.2.3.1.3.17 Lower

class LowerCls

Lower commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.clone()
```

Subgroups

6.2.3.1.3.18 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPpower:GAP<GapChannel>:MANual:LOWer:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:GAP<gap>:MANual:LOWer:ABSolute
value: float = driver.calculate.limit.acPower.gap.manual.lower.absolute.get(sb_
↳ gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

limit: No help available

set(*sb_gaps: SubBlockGaps*, *limit: float*, *window=Window.Default*, *limitIx=LimitIx.Default*, *gapChannel=GapChannel.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:MANual:LOWer:ABSolute
driver.calculate.limit.acPower.gap.manual.lower.absolute.set(sb_gaps = enums.
↳SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.absolute.clone()
```

Subgroups

6.2.3.1.3.19 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:MANual:LOWer:ABSolute:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>:MANual:LOWer:ABSolute:STATe
value: bool = driver.calculate.limit.acPower.gap.manual.lower.absolute.state.
↳ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx,
↳ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

state: No help available

set(*sb_gaps*: SubBlockGaps, *state*: bool, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>:MANual:LOWer:ABSolute:STATe
driver.calculate.limit.acPower.gap.manual.lower.absolute.state.set(sb_gaps =
↳ enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default, limitIx,
↳ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

6.2.3.1.3.20 Aclr

class AclrCls

Aclr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.aclr.clone()
```

Subgroups

6.2.3.1.3.21 Relative

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowEr:GAP<GapChannel>:MANual:LOWer:ACLR:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:GAP<gap>:MANual:LOWer:ACLR[:RELative]
value: float = driver.calculate.limit.acPower.gap.manual.lower.aclr.relative.
↪ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx_
↪ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

limit: No help available

set(sb_gaps: SubBlockGaps, limit: float, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None


```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>:MANual:LOWer:ACLR[:RELative]
driver.calculate.limit.acPower.gap.manual.lower.aclr.relative.set(sb_gaps =
↳enums.SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.aclr.relative.clone()
```

Subgroups

6.2.3.1.3.22 State

SCPI Commands

```
CALCulate<Window>:LIMIT<LimitIx>:ACPower:GAP<GapChannel>:MANual:LOWer:ACLR:RELative:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>
↳:MANual:LOWer:ACLR[:RELative]:STATE
value: bool = driver.calculate.limit.acPower.gap.manual.lower.aclr.relative.
↳state.get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(*sb_gaps*: SubBlockGaps, *state*: bool, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>
↳:MANual:LOWer:ACLR[:RELative]:STATE
driver.calculate.limit.acPower.gap.manual.lower.aclr.relative.state.set(sb_gaps,
↳= enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.2.3.1.3.23 Cacldr

class CacldrCls

Cacldr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.caclr.clone()
```

Subgroups

6.2.3.1.3.24 Relative

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:MANual:LOWer:CACLr:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:MANual:LOWer[:CACLr][:RELative]
value: float = driver.calculate.limit.acPower.gap.manual.lower.caclr.relative.
↳ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

limit: No help available

set(sb_gaps: SubBlockGaps, limit: float, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:MANual:LOWer[:CACLr][:RELative]
driver.calculate.limit.acPower.gap.manual.lower.caclr.relative.set(sb_gaps =
↳ enums.SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.lower.caclr.relative.clone()
```

Subgroups

6.2.3.1.3.25 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>
↳:MANual:LOWer[:CACLr][:RELative]:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default,
    gapChannel=GapChannel.Default) → bool
```

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>
↳:MANual:LOWer[:CACLr][:RELative]:STATe
value: bool = driver.calculate.limit.acPower.gap.manual.lower.caclr.relative.
↳state.get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(*sb_gaps: SubBlockGaps, state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:GAP<gap>
↳:MANual:LOWer[:CAClr][:RELative]:STATe
driver.calculate.limit.acPower.gap.manual.lower.caclr.relative.state.set(sb_
↳gaps = enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.2.3.1.3.26 Upper**class UpperCls**

Upper commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.clone()
```

Subgroups

6.2.3.1.3.27 Absolute

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:MANual:UPPer:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:MANual:UPPer:ABSolute
value: float = driver.calculate.limit.acPower.gap.manual.upper.absolute.get(sb_
↳ gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

limit: No help available

set(*sb_gaps*: SubBlockGaps, *limit*: float, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>:MANual:UPPer:ABSolute
driver.calculate.limit.acPower.gap.manual.upper.absolute.set(sb_gaps = enums.
↳ SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.absolute.clone()
```

Subgroups**6.2.3.1.3.28 State****SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:ACPowEr:GAP<GapChannel>:MANual:UPPer:ABSolute:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:GAP<gap>:MANual:UPPer:ABSolute:STATe
value: bool = driver.calculate.limit.acPower.gap.manual.upper.absolute.state.
↪ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx.
↪ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

```
set(sb_gaps: SubBlockGaps, state: bool, window=Window.Default, limitIx=LimitIx.Default,
    gapChannel=GapChannel.Default) → None
```

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:GAP<gap>:MANual:UPPer:ABSolute:STATe
driver.calculate.limit.acPower.gap.manual.upper.absolute.state.set(sb_gaps =
↳ enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default, limitIx.
↳ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

6.2.3.1.3.29 Aclr

class AclrCls

Aclr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.aclr.clone()
```

Subgroups

6.2.3.1.3.30 Relative

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowEr:GAP<GapChannel>:MANual:UPPer:ACLR:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

```
get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default,
    gapChannel=GapChannel.Default) → float
```



```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>:MANual:UPPer:ACLR[:RELative]
value: float = driver.calculate.limit.acPower.gap.manual.upper.aclr.relative.
↳ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

return

limit: No help available

set(sb_gaps: SubBlockGaps, limit: float, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:ACPower:GAP<gap>:MANual:UPPer:ACLR[:RELative]
driver.calculate.limit.acPower.gap.manual.upper.aclr.relative.set(sb_gaps =
↳ enums.SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳ repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.aclr.relative.clone()
```

Subgroups

6.2.3.1.3.31 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPoweR:GAP<GapChannel>:MANual:UPPer:ACLR:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>
↳:MANual:UPPer:ACLR[:RELative]:STATe
value: bool = driver.calculate.limit.acPower.gap.manual.upper.aclr.relative.
↳state.get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(sb_gaps: SubBlockGaps, state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPoweR:GAP<gap>
↳:MANual:UPPer:ACLR[:RELative]:STATe
driver.calculate.limit.acPower.gap.manual.upper.aclr.relative.state.set(sb_gaps,
↳enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.2.3.1.3.32 CacIrr**class CacIrrCls**

CacIrr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.cacIrr.clone()
```

Subgroups**6.2.3.1.3.33 Relative****SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:ACPpower:GAP<GapChannel>:MANual:UPPer:CACLr:RELative
```

class RelativeCls

Relative commands group definition. 2 total commands, 1 Subgroups, 1 group commands

```
get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default,
    gapChannel=GapChannel.Default) → float
```

```
# SCPI: CALCulate<n>:LIMit<li>:ACPpower:GAP<gap>:MANual:UPPer[:CACLr][:RELative]
value: float = driver.calculate.limit.acPower.gap.manual.upper.cacIrr.relative.
↪ get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default, limitIx_
↪ = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

limit: No help available

set(*sb_gaps*: SubBlockGaps, *limit*: float, *window*=Window.Default, *limitIx*=LimitIx.Default, *gapChannel*=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPower:GAP<gap>:MANual:UPPer[:CAClr][:RELative]
driver.calculate.limit.acPower.gap.manual.upper.caclr.relative.set(sb_gaps =
↳enums.SubBlockGaps.AB, limit = 1.0, window = repcap.Window.Default, limitIx =
↳repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.acPower.gap.manual.upper.caclr.relative.clone()
```

Subgroups

6.2.3.1.3.34 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACPowEr:GAP<GapChannel>
↳:MANual:UPPer:CACLr:RELative:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:GAP<gap>
↳:MANual:UPPer[:CACLr][:RELative]:STATe
value: bool = driver.calculate.limit.acPower.gap.manual.upper.caclr.relative.
↳state.get(sb_gaps = enums.SubBlockGaps.AB, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(sb_gaps: SubBlockGaps, state: bool, window=Window.Default, limitIx=LimitIx.Default, gapChannel=GapChannel.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ACPowEr:GAP<gap>
↳:MANual:UPPer[:CACLr][:RELative]:STATe
driver.calculate.limit.acPower.gap.manual.upper.caclr.relative.state.set(sb_
↳gaps = enums.SubBlockGaps.AB, state = False, window = repcap.Window.Default,
↳limitIx = repcap.LimitIx.Default, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Gap’)

6.2.3.1.4 State**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:ACPowEr:STATe
--

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:ACPowEr[:STATe] value: bool = driver.calculate.limit.acPower.state.get(window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

state: No help available

set(state: bool, window=Window.Default, limitIx=LimitIx.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:ACPowEr[:STATe] driver.calculate.limit.acPower.state.set(state = False, window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.2 Active

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ACTive
value: float = driver.calculate.limit.active.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command queries the names of all active limit lines.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit_lines: String containing the names of all active limit lines in alphabetical order.

6.2.3.3 Clear

class ClearCls

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.clear.clone()
```

Subgroups

6.2.3.3.1 Immediate

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:CLEar:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CLEar[:IMMediate]
driver.calculate.limit.clear.immediate.set(window = repcap.Window.Default,
↪limitIx = repcap.LimitIx.Default)
```

This command deletes the result of the current limit check. The command works on all limit lines in all measurement windows at the same time.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

set_with_opc(*window=Window.Default, limitIx=LimitIx.Default, opc_timeout_ms: int = -1*) → None

6.2.3.4 Comment

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:COMMENT
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → str

```
# SCPI: CALCulate<n>:LIMit<li>:COMMENT
value: str = driver.calculate.limit.comment.get(window = repcap.Window.Default,
↪limitIx = repcap.LimitIx.Default)
```

This command defines a comment for a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

comment: String containing the description of the limit line.

set(*comment: str, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:COMMENT
driver.calculate.limit.comment.set(comment = '1', window = repcap.Window.
↪Default, limitIx = repcap.LimitIx.Default)
```

This command defines a comment for a limit line.

param comment

String containing the description of the limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5 Control**class ControlCls**

Control commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.control.clone()
```

Subgroups**6.2.3.5.1 Data****SCPI Commands**

```
CALCulate<Window>:LIMit<LimitIx>:CONTrol:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol[:DATA]
value: List[float] = driver.calculate.limit.control.data.get(window = repcap.
↪Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the horizontal definition points of a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit_line_points: Variable number of x-axis values. Note that the number of horizontal values has to be the same as the number of vertical values set with method RsFswp.Calculate.Limit.Lower.Data.set or method RsFswp.Calculate.Limit.Upper.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: HZ

set(*limit_line_points*: List[float], *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol[:DATA]
driver.calculate.limit.control.data.set(limit_line_points = [1.1, 2.2, 3.3],
↪window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the horizontal definition points of a limit line.

param limit_line_points

Variable number of x-axis values. Note that the number of horizontal values has to be the same as the number of vertical values set with method RsFswp.Calculate.Limit.Lower.Data.set or method RsFswp.Calculate.Limit.Upper.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5.2 Domain

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:CONTrol:DOMain
```

class DomainCls

Domain commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default, *limitIx*=LimitIx.Default) → SpanSetting

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:DOMain
value: enums.SpanSetting = driver.calculate.limit.control.domain.get(window =
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the domain of the limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

span_setting: FREQuency | TIME FREQuency For limit lines that apply to a range of frequencies. TIME For limit lines that apply to a period of time. CURRent For limit lines that apply to a range of currents. VOLTage For limit lines that apply to a range of voltages.

set(span_setting: SpanSetting, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:DOMain
driver.calculate.limit.control.domain.set(span_setting = enums.SpanSetting.
↪FREquency, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the domain of the limit line.

param span_setting

FREquency | TIME FREquency For limit lines that apply to a range of frequencies. TIME For limit lines that apply to a period of time. CURRENT For limit lines that apply to a range of currents. VOLTage For limit lines that apply to a range of voltages.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5.3 Mode

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:CONTrol:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → ReferenceMode

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:MODE
value: enums.ReferenceMode = driver.calculate.limit.control.mode.get(window = ↪
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the horizontal limit line scaling.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

mode: ABSolute Limit line is defined by absolute physical values (Hz or s) . RELative Limit line is defined by relative values related to the center frequency (frequency domain) or the left diagram border (time domain) .

set(mode: ReferenceMode, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:MODE
driver.calculate.limit.control.mode.set(mode = enums.ReferenceMode.ABSolute, ↪
↪window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the horizontal limit line scaling.

param mode

ABSolute Limit line is defined by absolute physical values (Hz or s) . RELative Limit line is defined by relative values related to the center frequency (frequency domain) or the left diagram border (time domain) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5.4 Offset

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:CONTrol:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:OFFSet
value: float = driver.calculate.limit.control.offset.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

offset: No help available

set(offset: float, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:OFFSet
driver.calculate.limit.control.offset.set(offset = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param offset

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5.5 Shift**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:CONTrol:SHIFt
--

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → float

<pre># SCPI: CALCulate<n>:LIMit:CONTrol:SHIFt value: float = driver.calculate.limit.control.shift.get(window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>

This command moves a complete limit line horizontally. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

distance: Numeric value. The unit depends on the scale of the x-axis. Unit: HZ

set(distance: float, window=Window.Default, limitIx=LimitIx.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:CONTrol:SHIFt driver.calculate.limit.control.shift.set(distance = 1.0, window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>
--

This command moves a complete limit line horizontally. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Numeric value. The unit depends on the scale of the x-axis. Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.5.6 Spacing

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:CONTrol:SPACing`

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → ScalingMode

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:SPACing
value: enums.ScalingMode = driver.calculate.limit.control.spacing.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects linear or logarithmic interpolation for the calculation of limit lines from one horizontal point to the next.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

interpol_mode: LINear | LOGarithmic

set(*interpol_mode: ScalingMode, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:CONTrol:SPACing
driver.calculate.limit.control.spacing.set(interpol_mode = enums.ScalingMode.
↳LINear, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects linear or logarithmic interpolation for the calculation of limit lines from one horizontal point to the next.

param interpol_mode

LINear | LOGarithmic

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.6 Copy

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:COPY

class CopyCls

Copy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → int

```
# SCPI: CALCulate<n>:LIMit<li>:COPY
value: int = driver.calculate.limit.copy.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command copies a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

line: 1 to 8 number of the new limit line name String containing the name of the limit line.

set(line: int, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:COPY
driver.calculate.limit.copy.set(line = 1, window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command copies a limit line.

param line

1 to 8 number of the new limit line name String containing the name of the limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.7 Spectrum<SubBlock>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.calculate.limit.spectrum.repcap_subBlock_get()
driver.calculate.limit.spectrum.repcap_subBlock_set(repcap.SubBlock.Nr1)
```

class SpectrumCls

Spectrum commands group definition. 9 total commands, 5 Subgroups, 0 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.spectrum.clone()
```

Subgroups

6.2.3.7.1 Limits

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:LIMits
```

class LimitsCls

Limits commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:LIMits
value: List[float] = driver.calculate.limit.spectrum.limits.get(window = ↵
↵repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↵SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘ESpectrum’)

return

max_1: No help available

set(max_1: List[float], window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:LIMits
driver.calculate.limit.espectrum.limits.set(max_1 = [1.1, 2.2, 3.3], window = ␣
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param max_1

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.2.3.7.2 Mode

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default) → AutoManualUserMode

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:MODE
value: enums.AutoManualUserMode = driver.calculate.limit.espectrum.mode.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, ␣
↪subBlock = repcap.SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

mode: No help available

set(mode: AutoManualUserMode, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:ESpectrum<sb>:MODE
driver.calculate.limit.espectrum.mode.set(mode = enums.AutoManualUserMode.AUTO,
↪ window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock =
↪ repcap.SubBlock.Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

6.2.3.7.3 Pclass<PowerClass>

RepCap Settings

```
# Range: Nr1 .. Nr30
rc = driver.calculate.limit.espectrum.pclass.repcap_powerClass_get()
driver.calculate.limit.espectrum.pclass.repcap_powerClass_set(repcap.PowerClass.Nr1)
```

class PclassCls

Pclass commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: PowerClass, default value after init: PowerClass.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.espectrum.pclass.clone()
```

Subgroups

6.2.3.7.3.1 Count

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:PCLass<PowerClass>:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:COUNT
value: float = driver.calculate.limit.espectrum.pclass.count.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↳SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

return

no_power_classes: No help available

set(*no_power_classes: float, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:COUNT
driver.calculate.limit.espectrum.pclass.count.set(no_power_classes = 1.0,
↳window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock =
↳repcap.SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param no_power_classes

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

6.2.3.7.3.2 Exclusive**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:PCLass<PowerClass>:EXCLUSIVE

class ExclusiveCls

Exclusive commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default*) → bool

<pre># SCPI: CALCulate<n>:LIMit:ESpectrum<sb>:PCLass<pc>[:EXCLUSIVE] value: bool = driver.calculate.limit.espectrum.pclass.exclusive.get(window = ↵ ↵repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap. ↵SubBlock.Default, powerClass = repcap.PowerClass.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

return

state: No help available

set(*state: bool, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default*) → None

<pre># SCPI: CALCulate<n>:LIMit:ESpectrum<sb>:PCLass<pc>[:EXCLUSIVE] driver.calculate.limit.espectrum.pclass.exclusive.set(state = False, window = ↵</pre>
--

(continues on next page)

(continued from previous page)

```
↪repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↪SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

6.2.3.7.3.3 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.espectrum.pclass.limit.clone()
```

Subgroups

6.2.3.7.3.4 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:PCLass<PowerClass>:LIMit:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default,
powerClass=PowerClass.Default) → LimitState
```

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:LIMit[:STATe]
value: enums.LimitState = driver.calculate.limit.espectrum.pclass.limit.state.
↪get(window = repcap.Window.Default, limitIx = repcap.LimitIx.Default, ↪
↪subBlock = repcap.SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pclass’)

return

state: No help available

set(state: LimitState, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:LIMit[:STATe]
driver.calculate.limit.espectrum.pclass.limit.state.set(state = enums.
↪LimitState.Absolute, window = repcap.Window.Default, limitIx = repcap.LimitIx.
↪Default, subBlock = repcap.SubBlock.Default, powerClass = repcap.PowerClass.
↪Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pclass’)

6.2.3.7.3.5 Maximum

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:PCLass<PowerClass>:MAXimum

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:MAXimum
value: float = driver.calculate.limit.espectrum.pclass.maximum.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↳SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

return

level: No help available

set(level: float, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:PCLass<pc>:MAXimum
driver.calculate.limit.espectrum.pclass.maximum.set(level = 1.0, window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.
↳SubBlock.Default, powerClass = repcap.PowerClass.Default)
```

No command help available

param level

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pclass’)

6.2.3.7.3.6 Minimum**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:PCLass<PowerClass>:MINimum

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default) → float

<pre># SCPI: CALCulate<n>:LIMit:ESpectrum<sb>:PCLass<pc>:MINimum value: float = driver.calculate.limit.espectrum.pclass.minimum.get(window = ↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap. ↳SubBlock.Default, powerClass = repcap.PowerClass.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pclass’)

return

level: No help available

set(level: float, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, powerClass=PowerClass.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:ESpectrum<sb>:PCLass<pc>:MINimum driver.calculate.limit.espectrum.pclass.minimum.set(level = 1.0, window = ↳repcap.Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap. ↳SubBlock.Default, powerClass = repcap.PowerClass.Default)</pre>
--

No command help available

param level

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'ESpectrum')

param powerClass

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pclass')

6.2.3.7.4 Restore**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:REStore
--

class RestoreCls

Restore commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:ESpectrum<sb>:REStore driver.calculate.limit.espectrum.restore.set(window = repcap.Window.Default, ↪ limitIx = repcap.LimitIx.Default, subBlock = repcap.SubBlock.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'ESpectrum')

set_with_opc(window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default, opc_timeout_ms: int = -1) → None

6.2.3.7.5 Value

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:ESpectrum<SubBlock>:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:VALue
value: float = driver.calculate.limit.espectrum.value.get(window = repcap.
↳Window.Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

power: No help available

set(*power: float, window=Window.Default, limitIx=LimitIx.Default, subBlock=SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:ESpectrum<sb>:VALue
driver.calculate.limit.espectrum.value.set(power = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param power

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.2.3.8 Fail

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:FAIL
```

class FailCls

Fail commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:FAIL
value: float = driver.calculate.limit.fail.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command queries the result of a limit check in the specified window. To get a valid result, you have to perform a complete measurement with synchronization to the end of the measurement before reading out the result. This is only possible for single measurement mode.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

result: 0 PASS 1 FAIL

6.2.3.9 Lower

class LowerCls

Lower commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.lower.clone()
```

Subgroups

6.2.3.9.1 Data

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer[:DATA]
value: List[float] = driver.calculate.limit.lower.data.get(window = repcap.
↳ Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of a lower limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

limit_line_points: Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

set(*limit_line_points: List[float], window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer[:DATA]
driver.calculate.limit.lower.data.set(limit_line_points = [1.1, 2.2, 3.3],
↳ window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of a lower limit line.

param limit_line_points

Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.9.2 Margin

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:MARGin
```

class MarginCls

Margin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:MARGin
value: float = driver.calculate.limit.lower.margin.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

margin: No help available

set(margin: float, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:MARGin
driver.calculate.limit.lower.margin.set(margin = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param margin

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.9.3 Mode

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → ReferenceMode

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:MODE
value: enums.ReferenceMode = driver.calculate.limit.lower.mode.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the vertical limit line scaling.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

mode: ABSolute Limit line is defined by absolute physical values. The unit is variable.
RELative Limit line is defined by relative values related to the reference level (dB) .

set(mode: *ReferenceMode*, window=*Window.Default*, limitIx=*LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:MODE
driver.calculate.limit.lower.mode.set(mode = enums.ReferenceMode.Absolute,
↪window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the vertical limit line scaling.

param mode

ABSolute Limit line is defined by absolute physical values. The unit is variable. REL-
ative Limit line is defined by relative values related to the reference level (dB) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cal-
culate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface
'Limit')

6.2.3.9.4 Offset

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*, limitIx=*LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:OFFSet
value: float = driver.calculate.limit.lower.offset.get(window = repcap.Window.
↪Default, limitIx = repcap.LimitIx.Default)
```

This command defines an offset for a complete lower limit line. Compared to shifting the limit line, an offset does not actually change the limit line definition points.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cal-
culate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface
'Limit')

return

offset: Numeric value. Unit: dB

set(*offset: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:OFFSet
driver.calculate.limit.lower.offset.set(offset = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command defines an offset for a complete lower limit line. Compared to shifting the limit line, an offset does not actually change the limit line definition points.

param offset

Numeric value. Unit: dB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.9.5 Shift

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:LOWer:SHIFt
```

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:SHIFt
value: float = driver.calculate.limit.lower.shift.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete lower limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

distance: Defines the distance that the limit line moves. Unit: DB

set(*distance: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:SHIFt
driver.calculate.limit.lower.shift.set(distance = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete lower limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Defines the distance that the limit line moves. Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.9.6 Spacing**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:LOWer:SPACing
--

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → ScalingMode

<pre># SCPI: CALCulate<n>:LIMit:LOWer:SPACing value: enums.ScalingMode = driver.calculate.limit.lower.spacing.get(window = ↵ ↵repcap.Window.Default, limitIx = repcap.LimitIx.Default)</pre>
--

This command selects linear or logarithmic interpolation for the calculation of a lower limit line from one horizontal point to the next.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

interpol_type: LINear | LOGarithmic

set(interpol_type: ScalingMode, window=Window.Default, limitIx=LimitIx.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:LOWer:SPACing driver.calculate.limit.lower.spacing.set(interpol_type = enums.ScalingMode. ↵LINear, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)</pre>
--

This command selects linear or logarithmic interpolation for the calculation of a lower limit line from one horizontal point to the next.

param interpol_type

LINear | LOGarithmic

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.9.7 State**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:LOWer:STATe
--

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → bool

<pre># SCPI: CALCulate<n>:LIMit:LOWer:STATe value: bool = driver.calculate.limit.lower.state.get(window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>
--

This command turns a lower limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, limitIx=LimitIx.Default) → None

<pre># SCPI: CALCulate<n>:LIMit:LOWer:STATe driver.calculate.limit.lower.state.set(state = False, window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>

This command turns a lower limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.9.8 Threshold

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:LOWer:THReshold`

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:THReshold
value: float = driver.calculate.limit.lower.threshold.get(window = repcap.
↳Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

threshold: No help available

set(*threshold: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:LOWer:THReshold
driver.calculate.limit.lower.threshold.set(threshold = 1.0, window = repcap.
↳Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param threshold

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.10 Name

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → str

```
# SCPI: CALCulate<n>:LIMit<li>:NAME
value: str = driver.calculate.limit.name.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command selects a limit line that already exists or defines a name for a new limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

name: String containing the limit line name.

set(name: str, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:NAME
driver.calculate.limit.name.set(name = '1', window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command selects a limit line that already exists or defines a name for a new limit line.

param name

String containing the limit line name.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.11 State

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → bool

```
# SCPI: CALCulate<n>:LIMIT<li>:STATE
value: bool = driver.calculate.limit.state.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command turns the limit check for a specific limit line on and off. To query the limit check result, use method **RsFswp.Calculate.Limit.Fail.get_**. Note that a new command exists to activate the limit check and define the trace to be checked in one step (see method **RsFswp.Calculate.Limit.Trace.Check.set**).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMIT<li>:STATE
driver.calculate.limit.state.set(state = False, window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default)
```

This command turns the limit check for a specific limit line on and off. To query the limit check result, use method **RsFswp.Calculate.Limit.Fail.get_**. Note that a new command exists to activate the limit check and define the trace to be checked in one step (see method **RsFswp.Calculate.Limit.Trace.Check.set**).

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.12 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.limit.trace.repcap_trace_get()
driver.calculate.limit.trace.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:TRACe<Trace>

class TraceCls

Trace commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, limitIx=LimitIx.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe<t>
value: float = driver.calculate.limit.trace.get(window = repcap.Window.Default,
↪ limitIx = repcap.LimitIx.Default, trace = repcap.Trace.Default)
```

This command links a limit line to one or more traces. Note that this command is maintained for compatibility reasons only. Limit lines no longer need to be assigned to a trace explicitly. The trace to be checked can be defined directly (as a suffix) in the new command to activate the limit check (see method RsFswp.Calculate.Limit.Trace.Check.set) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

trace_limit: No help available

set(trace_limit: float, window=Window.Default, limitIx=LimitIx.Default, trace=Trace.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe<t>
driver.calculate.limit.trace.set(trace_limit = 1.0, window = repcap.Window.
↪ Default, limitIx = repcap.LimitIx.Default, trace = repcap.Trace.Default)
```

This command links a limit line to one or more traces. Note that this command is maintained for compatibility reasons only. Limit lines no longer need to be assigned to a trace explicitly. The trace to be checked can be defined directly (as a suffix) in the new command to activate the limit check (see method RsFswp.Calculate.Limit.Trace.Check.set) .

param trace_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.trace.clone()
```

Subgroups

6.2.3.12.1 Check

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:TRACe<Trace>:CHECK
```

class CheckCls

Check commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default, trace=Trace.Default) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe<t>:CHECK
value: bool = driver.calculate.limit.trace.check.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default, trace = repcap.Trace.Default)
```

This command turns the limit check for a specific trace on and off. To query the limit check result, use method RsFswp. **Calculate.Limit.Fail.get_**.

INTRO_CMD_HELP: Note that this command replaces the two commands from previous signal and spectrum analyzers (which are still supported, however) :

- CALCulate<n>:LIMit:TRACe<t>
- method RsFswp.Calculate.Limit.State.set

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, limitIx=LimitIx.Default, trace=Trace.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:TRACe<t>:CHECK
driver.calculate.limit.trace.check.set(state = False, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default, trace = repcap.Trace.Default)
```

This command turns the limit check for a specific trace on and off. To query the limit check result, use method RsFswp. **Calculate.Limit.Fail.get_**.

INTRO_CMD_HELP: Note that this command replaces the two commands from previous signal and spectrum analyzers (which are still supported, however) :

- CALCulate<n>:LIMit:TRACe<t>
- method RsFswp.Calculate.Limit.State.set

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.2.3.13 Unit

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → PowerUnitC

```
# SCPI: CALCulate<n>:LIMit<li>:UNIT
value: enums.PowerUnitC = driver.calculate.limit.unit.get(window = repcap.
↳ Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the unit of a limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

unit: If you select a dB-based unit for the limit line, the command automatically turns the limit line into a relative limit line.

set(unit: PowerUnitC, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UNIT
driver.calculate.limit.unit.set(unit = enums.PowerUnitC.A, window = repcap.
↳ Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the unit of a limit line.

param unit

If you select a dB-based unit for the limit line, the command automatically turns the limit line into a relative limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.14 Upper

class UpperCls

Upper commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.limit.upper.clone()
```

Subgroups

6.2.3.14.1 Data

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UPPer:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → List[float]

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer[:DATA]
value: List[float] = driver.calculate.limit.upper.data.get(window = repcap.
↳ Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of an upper limit line.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

limit_line_points: Variable number of level values. Note that the number of vertical

values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

set(*limit_line_points*: List[float], *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer[:DATA]
driver.calculate.limit.upper.data.set(limit_line_points = [1.1, 2.2, 3.3],
↪window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command defines the vertical definition points of an upper limit line.

param limit_line_points

Variable number of level values. Note that the number of vertical values has to be the same as the number of horizontal values set with method RsFswp.Calculate.Limit.Control.Data.set. If not, the R&S FSWP either adds missing values or ignores surplus values. Unit: DBM

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.14.2 Margin

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UPPer:MARGIN
```

class MarginCls

Margin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=Window.Default, *limitIx*=LimitIx.Default) → float

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:MARGIN
value: float = driver.calculate.limit.upper.margin.get(window = repcap.Window.
↪Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

margin: No help available

set(*margin*: float, *window*=Window.Default, *limitIx*=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:MARGin
driver.calculate.limit.upper.margin.set(margin = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param margin

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.14.3 Mode

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UPPer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → ReferenceMode

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:MODE
value: enums.ReferenceMode = driver.calculate.limit.upper.mode.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the vertical limit line scaling.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

mode: ABSolute Limit line is defined by absolute physical values. The unit is variable. RELative Limit line is defined by relative values related to the reference level (dB) .

set(mode: ReferenceMode, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:MODE
driver.calculate.limit.upper.mode.set(mode = enums.ReferenceMode.ABSolute,
↳window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects the vertical limit line scaling.

param mode

ABSolute Limit line is defined by absolute physical values. The unit is variable. RELative Limit line is defined by relative values related to the reference level (dB) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.14.4 Offset**SCPI Commands**

CALCulate<Window>:LIMit<LimitIx>:UPPer:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

<pre># SCPI: CALCulate<n>:LIMit:UPPer:OFFSet value: float = driver.calculate.limit.upper.offset.get(window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

offset: No help available

set(*offset: float, window=Window.Default, limitIx=LimitIx.Default*) → None

<pre># SCPI: CALCulate<n>:LIMit:UPPer:OFFSet driver.calculate.limit.upper.offset.set(offset = 1.0, window = repcap.Window. ↳Default, limitIx = repcap.LimitIx.Default)</pre>
--

No command help available

param offset

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.14.5 Shift

SCPI Commands

`CALCulate<Window>:LIMit<LimitIx>:UPPer:SHIFt`

class ShiftCls

Shift commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:SHIFt
value: float = driver.calculate.limit.upper.shift.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete upper limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

return

distance: Defines the distance that the limit line moves.

set(*distance: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:SHIFt
driver.calculate.limit.upper.shift.set(distance = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command moves a complete upper limit line vertically. Compared to defining an offset, this command actually changes the limit line definition points by the value you define.

param distance

Defines the distance that the limit line moves.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.2.3.14.6 Spacing

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:UPPer:SPACing

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, limitIx=LimitIx.Default) → ScalingMode

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:SPACing
value: enums.ScalingMode = driver.calculate.limit.upper.spacing.get(window =
↳repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects linear or logarithmic interpolation for the calculation of an upper limit line from one horizontal point to the next.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

interpol_type: LINear | LOGarithmic

set(interpol_type: ScalingMode, window=Window.Default, limitIx=LimitIx.Default) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:SPACing
driver.calculate.limit.upper.spacing.set(interpol_type = enums.ScalingMode.
↳LINear, window = repcap.Window.Default, limitIx = repcap.LimitIx.Default)
```

This command selects linear or logarithmic interpolation for the calculation of an upper limit line from one horizontal point to the next.

param interpol_type

LINear | LOGarithmic

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.14.7 State

SCPI Commands

CALCulate<Window>:LIMit<LimitIx>:UPPer:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → bool

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:STATe
value: bool = driver.calculate.limit.upper.state.get(window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command turns an upper limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:STATe
driver.calculate.limit.upper.state.set(state = False, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

This command turns an upper limit line on and off. Before you can use the command, you have to select a limit line with method RsFswp.Calculate.Limit.Name.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.3.14.8 Threshold

SCPI Commands

```
CALCulate<Window>:LIMit<LimitIx>:UPPer:THReshold
```

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, limitIx=LimitIx.Default*) → float

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:THReshold
value: float = driver.calculate.limit.upper.threshold.get(window = repcap.
↳Window.Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

limit: No help available

set(*limit: float, window=Window.Default, limitIx=LimitIx.Default*) → None

```
# SCPI: CALCulate<n>:LIMit<li>:UPPer:THReshold
driver.calculate.limit.upper.threshold.set(limit = 1.0, window = repcap.Window.
↳Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.2.4 Marker<Marker>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.calculate.marker.repcap_marker_get()
driver.calculate.marker.repcap_marker_set(repcap.Marker.Nr1)
```

class MarkerCls

Marker commands group definition. 131 total commands, 15 Subgroups, 0 group commands Repeated Capability: Marker, default value after init: Marker.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.clone()
```

Subgroups

6.2.4.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer:AOff
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer:AOff
driver.calculate.marker.aoff.set(window = repcap.Window.Default)
```

This command turns off all markers.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.2 Count

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:COUNT
```

class CountCls

Count commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:COUNT
value: bool = driver.calculate.marker.count.get(window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:COUNT
driver.calculate.marker.count.set(state = False, window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.count.clone()
```

Subgroups

6.2.4.2.1 Frequency

SCPI Commands

CALCulate<Window>:MARKer<Marker>:COUNT:FREQuency

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:COUNT:FREQuency
value: float = driver.calculate.marker.count.frequency.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

frequency: No help available

6.2.4.2.2 Resolution

SCPI Commands

CALCulate<Window>:MARKer<Marker>:COUNT:RESolution

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:COUNT:RESolution
value: float = driver.calculate.marker.count.resolution.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

resolution: No help available

set(resolution: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:COUNT:RESolution
driver.calculate.marker.count.resolution.set(resolution = 1.0, window = repcap.
↪Window.Default, marker = repcap.Marker.Default)
```

No command help available

param resolution

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3 Function

class FunctionCls

Function commands group definition. 90 total commands, 17 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.clone()
```

Subgroups

6.2.4.3.1 Ademod

class AdemodCls

Ademod commands group definition. 12 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.clone()
```

Subgroups

6.2.4.3.1.1 Afrequency

class AfrequencyCls

Afrequency commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.afrequency.clone()
```

Subgroups

6.2.4.3.1.2 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.afrequency.result.repcap_trace_get()
driver.calculate.marker.function.ademod.afrequency.result.repcap_trace_set(repcap.Trace.
↳Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:ADEMod:AFRequency:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:ADEMod:AFRequency[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.afrequency.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default, trace =
↳repcap.Trace.Default)
```

This command queries the modulation (audio) frequency for the demodulation method in the specified window.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Result')

return

mod_freq: Modulation frequency in Hz.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.afrequency.result.clone()
```

6.2.4.3.1.3 Am**class AmCls**

Am commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.am.clone()
```

Subgroups**6.2.4.3.1.4 Result<Trace>****RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.am.result.repcap_trace_get()
driver.calculate.marker.function.ademod.am.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:AM:RESult<Trace>
```

class ResultCls

Result commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

```
get(meas_type: AdemMeasType, window=Window.Default, marker=Marker.Default, trace=Trace.Default)
    → float
```

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:AM[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.am.result.get(meas_type,
↪enums.AdemMeasType.MIDDLE, window = repcap.Window.Default, marker = repcap.
↪Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces, thus the window is irrelevant.

param meas_type

PPEak | MPEak | MIDDLE | RMS PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks $\pm PK/2$ RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

meas_type_result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.am.result.clone()
```

Subgroups

6.2.4.3.1.5 Relative

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:AM:RESult<Trace>:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(meas_type: AdemMeasType, window=Window.Default, marker=Marker.Default, trace=Trace.Default)
→ float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:AM[:RESult<t>]:RELative
value: float = driver.calculate.marker.function.ademod.am.result.relative.
↳ get(meas_type = enums.AdemMeasType.MIDDLE, window = repcap.Window.Default,
↳ marker = repcap.Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current relative value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces. The unit of the results depends on the method RsFswp.Configure.Ademod.Results.Unit.set setting.

param meas_type

PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks \pm PK/2 RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Result')

return

meas_type_result: No help available

6.2.4.3.1.6 Carrier**class CarrierCls**

Carrier commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.carrier.clone()
```

Subgroups**6.2.4.3.1.7 Result<Trace>****RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.carrier.result.repcap_trace_get()
driver.calculate.marker.function.ademod.carrier.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:ADEMod:CARRier:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:ADEMod:CARRier[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.carrier.result.
↳ get(window = repcap.Window.Default, marker = repcap.Marker.Default, trace =
↳ repcap.Trace.Default)
```

This command queries the carrier power, which is determined from the Clr/Write data.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

cpower: Power of the carrier without modulation in dBm.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.carrier.result.clone()
```

6.2.4.3.1.8 Distortion

class DistortionCls

Distortion commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.distortion.clone()
```

Subgroups

6.2.4.3.1.9 Write

class WriteCls

Write commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.distortion.write.clone()
```

Subgroups

6.2.4.3.1.10 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.distortion.write.result.repcap_trace_get()
driver.calculate.marker.function.ademod.distortion.write.result.repcap_trace_set(repcap.
↳Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:ADEMod:DISTortion:WRITe:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:ADEMod:DISTortion[:WRITe]:RESult<t>
value: float = driver.calculate.marker.function.ademod.distortion.write.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default, trace =
↳repcap.Trace.Default)
```

This command queries the result of the modulation distortion measurement in the specified window for the specified trace. Note that this value is only calculated if an AF Spectrum window is displayed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Result')

return

distort: numeric value Modulation distortion in percent. Unit: %

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.distortion.write.result.clone()
```

6.2.4.3.1.11 Fm

class FmCls

Fm commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.fm.clone()
```

Subgroups

6.2.4.3.1.12 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.fm.result.repcap_trace_get()
driver.calculate.marker.function.ademod.fm.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:FM:RESult<Trace>
```

class ResultCls

Result commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

```
get(meas_type: AdemMeasType, window=Window.Default, marker=Marker.Default, trace=Trace.Default)
→ float
```

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:FM[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.fm.result.get(meas_type,
↪= enums.AdemMeasType.MIDDLE, window = repcap.Window.Default, marker = repcap.
↪Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces, thus the window is irrelevant.

param meas_type

PPEak | MPEak | MIDDLE | RMS PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks \pm PK/2 RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

meas_type_result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.fm.result.clone()
```

Subgroups**6.2.4.3.1.13 Relative****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:FM:RESult<Trace>:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get(meas_type: AdemMeasType, window=Window.Default, marker=Marker.Default, trace=Trace.Default)
→ float
```

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:FM[:RESult<t>]:RELative
value: float = driver.calculate.marker.function.ademod.fm.result.relative.
↳ get(meas_type = enums.AdemMeasType.MIDDLE, window = repcap.Window.Default,
↳ marker = repcap.Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current relative value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces. The unit of the results depends on the method RsFswp.Configure.Ademod.Results.Unit.set setting.

param meas_type

PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks \pm PK/2 RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Result')

return

meas_type_result: No help available

6.2.4.3.1.14 FreqError

class FreqErrorCls

FreqError commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.freqError.clone()
```

Subgroups

6.2.4.3.1.15 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.freqError.result.repcap_trace_get()
driver.calculate.marker.function.ademod.freqError.result.repcap_trace_set(repcap.Trace.
↳Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:ADEMod:FERRor:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:ADEMod:FERRor[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.freqError.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default, trace =
↳repcap.Trace.Default)
```

This command queries the carrier offset (= frequency error) for FM and PM demodulation. The carrier offset is determined from the current measurement data (CLR/WRITE) . The modulation is removed using low pass filtering.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Result')

return

carr_offset: The deviation of the calculated carrier frequency to the ideal carrier frequency in Hz.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.freqError.result.clone()
```

6.2.4.3.1.16 Pm**class PmCls**

Pm commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.pm.clone()
```

Subgroups**6.2.4.3.1.17 Result<Trace>****RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.pm.result.repcap_trace_get()
driver.calculate.marker.function.ademod.pm.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:PM:RESult<Trace>
```

class ResultCls

Result commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(*meas_type*: *AdemMeasType*, *window*=*Window.Default*, *marker*=*Marker.Default*, *trace*=*Trace.Default*)
→ float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:PM[:RESult<t>]
value: float = driver.calculate.marker.function.ademod.pm.result.get(meas_type,
↪= enums.AdemMeasType.MIDDLE, window = repcap.Window.Default, marker = repcap.
↪Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces, thus the window is irrelevant.

param meas_type

PPEak | MPEak | MIDDLE | RMS PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks \pm PK/2 RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

meas_type_result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.pm.result.clone()
```

Subgroups

6.2.4.3.1.18 Relative

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:PM:RESult<Trace>:RELative
```

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*meas_type*: *AdemMeasType*, *window*=*Window.Default*, *marker*=*Marker.Default*, *trace*=*Trace.Default*)
→ float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:PM[:RESult<t>]:RELative
value: float = driver.calculate.marker.function.ademod.pm.result.relative.
↪get(meas_type = enums.AdemMeasType.MIDDLE, window = repcap.Window.Default, ↪
↪marker = repcap.Marker.Default, trace = repcap.Trace.Default)
```

This command queries the current relative value of the demodulated signal for the specified trace (as displayed in the ‘Result Summary’ in manual operation) . Note that all windows with the same evaluation method have the same traces. The unit of the results depends on the method RsFswp.Configure.Ademod.Results.Unit.set setting.

param meas_type

PPEak Postive peak (+PK) MPEak | NPEak Negative peak (-PK) MIDDLE Average of positive and negative peaks \pm PK/2 RMS Root mean square value

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

meas_type_result: No help available

6.2.4.3.1.19 Sinad

class SinadCls

Sinad commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.sinad.clone()
```

Subgroups

6.2.4.3.1.20 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.sinad.result.repcap_trace_get()
driver.calculate.marker.function.ademod.sinad.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCtion:ADEMod:SINad:RESult<Trace>`

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:ADEMod:SINad:RESult<t>
value: float = driver.calculate.marker.function.ademod.sinad.result.get(window,
↳= repcap.Window.Default, marker = repcap.Marker.Default, trace = repcap.Trace.
↳Default)
```

This command queries the result of the signal-to-noise-and-distortion (SINAD) measurement in the specified window for the specified trace. Note that this value is only calculated if an AF Spectrum window is displayed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

sinad: The signal-to-noise-and-distortion ratio in dB.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.sinad.result.clone()
```

6.2.4.3.1.21 Thd

class ThdCls

Thd commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.thd.clone()
```

Subgroups

6.2.4.3.1.22 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.ademod.thd.result.repcap_trace_get()
driver.calculate.marker.function.ademod.thd.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:ADEMod:THD:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:ADEMod:THD:RESult<t>
value: float = driver.calculate.marker.function.ademod.thd.result.get(window =
↳ repcap.Window.Default, marker = repcap.Marker.Default, trace = repcap.Trace.
↳ Default)
```

This command queries the result of the total harmonic distortion (THD) measurement in the specified window. Note that this value is only calculated if an AF Spectrum window is displayed.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

thd: Total harmonic distortion of the demodulated signal in dB.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ademod.thd.result.clone()
```

6.2.4.3.2 AfPhase

class AfPhaseCls

AfPhase commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.afPhase.clone()
```

Subgroups

6.2.4.3.2.1 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:AFPHase:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:AFPHase:RESult
value: float = driver.calculate.marker.function.afPhase.result.get(window = ↩
↩repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

af_phase: No help available

6.2.4.3.2.2 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:AFPHase:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:AFPHase[:STATe]
value: bool = driver.calculate.marker.function.afPhase.state.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:AFPHase[:STATe]
driver.calculate.marker.function.afPhase.state.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.3 Bpower

class BpowerCls

Bpower commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.bpower.clone()
```

Subgroups

6.2.4.3.3.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:BPOWer:AOff
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:BPOWer:AOff
driver.calculate.marker.function.bpower.aoff.set(window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.3.2 Mode

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:BPOWer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → MarkerMode

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer:MODE
value: enums.MarkerMode = driver.calculate.marker.function.bpower.mode.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

mode: No help available

set(*mode: MarkerMode, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer:MODE
driver.calculate.marker.function.bpower.mode.set(mode = enums.MarkerMode.
↪DENSity, window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.3.3 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:BPOWer:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer:RESult
value: float = driver.calculate.marker.function.bpower.result.get(window =
↪repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

power: No help available

6.2.4.3.3.4 Span

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:BPOWer:SPAN`

class SpanCls

Span commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window*=*Window.Default*, *marker*=*Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer:SPAN
value: float = driver.calculate.marker.function.bpower.span.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

span: No help available

set(*span*: float, *window*=*Window.Default*, *marker*=*Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer:SPAN
driver.calculate.marker.function.bpower.span.set(span = 1.0, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param span

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.3.5 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:BPOWer:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer[:STATe]
value: bool = driver.calculate.marker.function.bpower.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:BPOWer[:STATe]
driver.calculate.marker.function.bpower.state.set(state = False, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.4 Center

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNction:CENTer`

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:CENTer
driver.calculate.marker.function.center.set(window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

This command matches the center frequency to the frequency of a marker. If you use the command in combination with a delta marker, that delta marker is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.5 Cstep

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNction:CSTep`

class CstepCls

Cstep commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:CSTep
driver.calculate.marker.function.cstep.set(window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.6 Fpeaks

class FpeaksCls

Fpeaks commands group definition. 8 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.fpeaks.clone()
```

Subgroups

6.2.4.3.6.1 Annotation

class AnnotationCls

Annotation commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.fpeaks.annotation.clone()
```

Subgroups

6.2.4.3.6.2 Label

class LabelCls

Label commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.fpeaks.annotation.label.clone()
```

Subgroups

6.2.4.3.6.3 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:FPEaks:ANNotation:LABel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:ANNotation:LABel[:STATe]
value: bool = driver.calculate.marker.function.fpeaks.annotation.label.state.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:ANNotation:LABel[:STATe]
driver.calculate.marker.function.fpeaks.annotation.label.state.set(state =
↳False, window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.6.4 Count

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEaks:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:COUNT
value: float = driver.calculate.marker.function.fpeaks.count.get(window =
↳recap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

number_of_peaks: No help available

6.2.4.3.6.5 Immediate**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEaks:IMMEDIATE
--

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(peaks: float, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks[:IMMEDIATE] driver.calculate.marker.function.fpeaks.immediate.set(peaks = 1.0, window = ↳repcap.Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param peaks

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.6.6 ListPy**class ListPyCls**

ListPy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.fpeaks.listPy.clone()
```

Subgroups

6.2.4.3.6.7 Size

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:FPEaks:LIST:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:FPEaks:LIST:SIZE
value: float = driver.calculate.marker.function.fpeaks.listPy.size.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

max_no_peaks: No help available

set(max_no_peaks: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:FPEaks:LIST:SIZE
driver.calculate.marker.function.fpeaks.listPy.size.set(max_no_peaks = 1.0,
↳window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param max_no_peaks

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.6.8 Sort

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNction:FPEaks:SORT

class SortCls

Sort commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → FpeaksSortMode

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:FPEaks:SORT
value: enums.FpeaksSortMode = driver.calculate.marker.function.fpeaks.sort.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

This command selects the order in which the results of a peak search are returned.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

sort_mode: X Sorts the peaks according to increasing position on the x-axis. Y Sorts the peaks according to decreasing position on the y-axis.

set(sort_mode: FpeaksSortMode, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:FPEaks:SORT
driver.calculate.marker.function.fpeaks.sort.set(sort_mode = enums.
↳FpeaksSortMode.X, window = repcap.Window.Default, marker = repcap.Marker.
↳Default)
```

This command selects the order in which the results of a peak search are returned.

param sort_mode

X Sorts the peaks according to increasing position on the x-axis. Y Sorts the peaks according to decreasing position on the y-axis.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.6.9 State

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEaks:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:STATe
value: bool = driver.calculate.marker.function.fpeaks.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:STATe
driver.calculate.marker.function.fpeaks.state.set(state = False, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.6.10 X

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEaks:X

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:X
value: float = driver.calculate.marker.function.fpeaks.x.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command queries the position of the peaks on the x-axis. The order depends on the sort order that has been set with method RsFswp.Calculate.Marker.Function.Fpeaks.Sort.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

peak_position: Position of the peaks on the x-axis. The unit depends on the measurement.

6.2.4.3.6.11 Y

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEaks:Y

class YCls

Y commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:FPEaks:Y
value: float = driver.calculate.marker.function.fpeaks.y.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

This command queries the position of the peaks on the y-axis. The order depends on the sort order that has been set with method RsFswp.Calculate.Marker.Function.Fpeaks.Sort.set.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

peak_position: Position of the peaks on the y-axis. The unit depends on the measurement.

6.2.4.3.7 Harmonics

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:PRESet`

class HarmonicsCls

Harmonics commands group definition. 6 total commands, 5 Subgroups, 1 group commands

preset(*window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:HARMonics:PRESet
driver.calculate.marker.function.harmonics.preset(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

preset_with_opc(*window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.harmonics.clone()
```

Subgroups

6.2.4.3.7.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.harmonics.bandwidth.clone()
```

Subgroups

6.2.4.3.7.2 Auto

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:HARMonics:BANDwidth:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:HARMonics:BANDwidth:AUTO
driver.calculate.marker.function.harmonics.bandwidth.auto.set(state = False,
↳window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.7.3 Distortion

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:HARMonics:DISToRTion
```

class DistortionCls

Distortion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Distortion_Pct: float: No parameter help available
- Distortion_Db: float: No parameter help available

get(*result: Optional[ResultTypeD] = None, window=Window.Default, marker=Marker.Default*) → GetStruct

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:HARMonics:DISortion
value: GetStruct = driver.calculate.marker.function.harmonics.distortion.
↪get(result = enums.ResultTypeD.TOTal, window = repcap.Window.Default, marker =
↪= repcap.Marker.Default)
```

No command help available

param result

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.4.3.7.4 ListPy

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:HARMonics:LIST
value: float = driver.calculate.marker.function.harmonics.listPy.get(window =
↪= repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

harmonics: No help available

6.2.4.3.7.5 Nharmonics

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:HARMonics:NHARmonics
```

class NharmonicsCls

Nharmonics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:HARMonics:NHARmonics
value: float = driver.calculate.marker.function.harmonics.nharmonics.get(window,
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

no_harmonics: No help available

set(*no_harmonics: float, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:HARMonics:NHARmonics
driver.calculate.marker.function.harmonics.nharmonics.set(no_harmonics = 1.0,
↳ window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param no_harmonics

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.7.6 State

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:HARMonics[:STATe]
value: bool = driver.calculate.marker.function.harmonics.state.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:HARMonics[:STATe]
driver.calculate.marker.function.harmonics.state.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.8 Mdepth

class MdepthCls

Mdepth commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.mdepth.clone()
```

Subgroups

6.2.4.3.8.1 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.marker.function.mdepth.result.repcap_trace_get()
driver.calculate.marker.function.mdepth.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:MDEPth:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(window=Window.Default, marker=Marker.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:MDEPth:RESult<t>
value: float = driver.calculate.marker.function.mdepth.result.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default, trace = repcap.Trace.
↵Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

modulation_depth: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.mdepth.result.clone()
```

6.2.4.3.8.2 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:MDEPth:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:MDEPth[:STATE]
value: bool = driver.calculate.marker.function.mdepth.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:MDEPth[:STATE]
driver.calculate.marker.function.mdepth.state.set(state = False, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.9 Msummary

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:MSUMmary

class MsummaryCls

Msummary commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MsummaryStruct

Response structure. Fields:

- Time_Offset: float: No parameter help available
- Meas_Time: float: No parameter help available
- Pulse_Period: float: No parameter help available
- Of_Pulses: float: No parameter help available

get(window=Window.Default, marker=Marker.Default) → MsummaryStruct

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:MSUMmary
value: MsummaryStruct = driver.calculate.marker.function.msummary.get(window =
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

structure: for return value, see the help for MsummaryStruct structure arguments.

set(time_offset: float, meas_time: float, pulse_period: float, of_pulses: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:MSUMmary
driver.calculate.marker.function.msummary.set(time_offset = 1.0, meas_time = 1.
↵0, pulse_period = 1.0, of_pulses = 1.0, window = repcap.Window.Default,
↵marker = repcap.Marker.Default)
```

No command help available

param time_offset

No help available

param meas_time

No help available

param pulse_period

No help available

param of_pulses

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.10 NdbDown

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:NDBDown

class NdbDownCls

NdbDown commands group definition. 6 total commands, 5 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown
value: float = driver.calculate.marker.function.ndbDown.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

distance: No help available

set(distance: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown
driver.calculate.marker.function.ndbDown.set(distance = 1.0, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param distance

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.ndbDown.clone()
```

Subgroups

6.2.4.3.10.1 Frequency

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:NDBDown:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:NDBDown:FREQuency
value: float = driver.calculate.marker.function.ndbDown.frequency.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

frequency: No help available

6.2.4.3.10.2 Qfactor

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:NDBDown:QFACTOR
```

class QfactorCls

Qfactor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:NDBDown:QFACTOR
value: float = driver.calculate.marker.function.ndbDown.qfactor.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

qfactor: No help available

6.2.4.3.10.3 Result

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:NDBDown:RESult

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown:RESult
value: float = driver.calculate.marker.function.ndbDown.result.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

distance: No help available

6.2.4.3.10.4 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:NDBDown:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown:STATe
value: bool = driver.calculate.marker.function.ndbDown.state.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown:STATE
driver.calculate.marker.function.ndbDown.state.set(state = False, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.10.5 Time

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:NDBDown:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Time_X_1: float: No parameter help available
- Time_X_2: float: No parameter help available

get(window=Window.Default, marker=Marker.Default) → GetStruct

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NDBDown:TIME
value: GetStruct = driver.calculate.marker.function.ndbDown.time.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.4.3.11 Noise

class NoiseCls

Noise commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.noise.clone()
```

Subgroups

6.2.4.3.11.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:NOISe:AOff
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:NOISe:AOff
driver.calculate.marker.function.noise.aoff.set(window = repcap.Window.Default,
↵marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.11.2 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:NOISe:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NOISe:RESult
value: float = driver.calculate.marker.function.noise.result.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

noise_level: No help available

6.2.4.3.11.3 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:NOISe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NOISe[:STATe]
value: bool = driver.calculate.marker.function.noise.state.get(window = repcap.
↵Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:NOISE[:STATe]
driver.calculate.marker.function.noise.state.set(state = False, window = repcap.
↪ Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.12 Pnoise

class PnoiseCls

Pnoise commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.pnoise.clone()
```

Subgroups

6.2.4.3.12.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:PNOise:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:PNOise:AOFF
driver.calculate.marker.function.pnoise.aoff.set(window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.12.2 Result**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:PNOise:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:PNOise:RESult
value: float = driver.calculate.marker.function.pnoise.result.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

phasenoise: No help available

6.2.4.3.12.3 State**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:PNOise:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:PNOise[:STATe]
value: bool = driver.calculate.marker.function.pnoise.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:PNOise[:STATE]
driver.calculate.marker.function.pnoise.state.set(state = False, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.13 Power<SubBlock>**RepCap Settings**

```
# Range: Nr1 .. Nr8
rc = driver.calculate.marker.function.power.repcap_subBlock_get()
driver.calculate.marker.function.power.repcap_subBlock_set(repcap.SubBlock.Nr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:PRESet
```

class PowerCls

Power commands group definition. 12 total commands, 6 Subgroups, 1 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

preset(standard: TechnologyStandardB, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:PRESet
driver.calculate.marker.function.power.preset(standard = enums.
↳TechnologyStandardB.AWLan, window = repcap.Window.Default, marker = repcap.
↳Marker.Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param standard

(enum or string) No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.power.clone()
```

Subgroups**6.2.4.3.13.1 Auto****class AutoCls**

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.power.auto.clone()
```

Subgroups**6.2.4.3.13.2 ListPy****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNction:POWer<SubBlock>:AUTO:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → str

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:POWer<sb>:AUTO:LIST
value: str = driver.calculate.marker.function.power.auto.listPy.get(window =
↪repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

list_py: No help available

6.2.4.3.13.3 Mode**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:MODE
--

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → TraceModeD

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:MODE value: enums.TraceModeD = driver.calculate.marker.function.power.mode. ↪ get(window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.SubBlock.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

mode: No help available

set(mode: TraceModeD, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:MODE driver.calculate.marker.function.power.mode.set(mode = enums.TraceModeD.MAXHold, ↪ window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.SubBlock.Default)</pre>
--

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.2.4.3.13.4 Result**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:RESult

class ResultCls

Result commands group definition. 4 total commands, 3 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → MarkerFunctionA

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult
value: enums.MarkerFunctionA = driver.calculate.marker.function.power.result.
↪ get(window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock_
↪ = repcap.SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

measurement: No help available

set(measurement: MarkerFunctionA, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult
driver.calculate.marker.function.power.result.set(measurement = enums.
↪ MarkerFunctionA.ACPower, window = repcap.Window.Default, marker = repcap.
↪ Marker.Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param measurement

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.power.result.clone()
```

Subgroups

6.2.4.3.13.5 Details

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:POWer<SubBlock>:RESult:DEtails
```

class DetailsCls

Details commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → MarkerFunctionA

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:POWer<sb>:RESult:DEtails
value: enums.MarkerFunctionA = driver.calculate.marker.function.power.result.
↳ details.get(window = repcap.Window.Default, marker = repcap.Marker.Default,↳
↳ subBlock = repcap.SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

return

measurement: No help available

set(measurement: MarkerFunctionA, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult:DEtails
driver.calculate.marker.function.power.result.details.set(measurement = enums.
↳MarkerFunctionA.ACPower, window = repcap.Window.Default, marker = repcap.
↳Marker.Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param measurement

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.2.4.3.13.6 Phz

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:RESult:PHZ
```

class PhzCls

Phz commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult:PHZ
value: bool = driver.calculate.marker.function.power.result.phz.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult:PHZ
driver.calculate.marker.function.power.result.phz.set(state = False, window =
↳ repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↳ SubBlock.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

6.2.4.3.13.7 Unit

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:RESult:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → PwrMeasUnit

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult:UNIT
value: enums.PwrMeasUnit = driver.calculate.marker.function.power.result.unit.
↳ get(window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock.
↳ = repcap.SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

return

measurement: No help available

set(*measurement: PwrMeasUnit, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:RESult:UNIT
driver.calculate.marker.function.power.result.unit.set(measurement = enums.
↳PwrMeasUnit.ABS, window = repcap.Window.Default, marker = repcap.Marker.
↳Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param measurement

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

6.2.4.3.13.8 Select

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*meas_type: MarkerFunctionB, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:SElect
driver.calculate.marker.function.power.select.set(meas_type = enums.
↳MarkerFunctionB.ACPower, window = repcap.Window.Default, marker = repcap.
↳Marker.Default, subBlock = repcap.SubBlock.Default)
```

No command help available

param meas_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.2.4.3.13.9 Standard**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNction:POWer<SubBlock>:STANdard:DELeTe

class StandardCls

Standard commands group definition. 3 total commands, 2 Subgroups, 1 group commands

delete(*standard: str, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:POWer<sb>:STANdard:DELeTe
driver.calculate.marker.function.power.standard.delete(standard = '1', window =
↳repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param standard

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.power.standard.clone()
```

Subgroups**6.2.4.3.13.10 Catalog****SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNction:POWer<SubBlock>:STANdard:CATalog

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:POWer<sb>:STANdard:CATalog
value: float = driver.calculate.marker.function.power.standard.catalog.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock_
↪= repcap.SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

standards: No help available

6.2.4.3.13.11 Save**SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:POWer<SubBlock>:STANdard:SAVE
```

class SaveCls

Save commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default) → str

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:POWer<sb>:STANdard:SAVE
value: str = driver.calculate.marker.function.power.standard.save.get(window =
↪repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

standard: No help available

set(*standard: str, window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>:STANdard:SAVE
driver.calculate.marker.function.power.standard.save.set(standard = '1', window_
↪= repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param standard

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.2.4.3.13.12 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:POWer<SubBlock>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default, subBlock=SubBlock.Default*) → OffState

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>[:STATe]
value: enums.OffState = driver.calculate.marker.function.power.state.get(window_
↪= repcap.Window.Default, marker = repcap.Marker.Default, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

state: No help available

set(state: *OffState*, window=*Window.Default*, marker=*Marker.Default*, subBlock=*SubBlock.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:POWer<sb>[:STATe]
driver.calculate.marker.function.power.state.set(state = enums.OffState.OFF,
↪window = repcap.Window.Default, marker = repcap.Marker.Default, subBlock =
↪repcap.SubBlock.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

6.2.4.3.14 Reference

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:REference

class ReferenceCls

Reference commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=*Window.Default*, marker=*Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:REference
driver.calculate.marker.function.reference.set(window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

This command matches the reference level to the power level of a marker. If you use the command in combination with a delta marker, that delta marker is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=*Window.Default*, marker=*Marker.Default*, opc_timeout_ms: *int* = -1) → None

6.2.4.3.15 Strack

class StrackCls

Strack commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.strack.clone()
```

Subgroups

6.2.4.3.15.1 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:STRack:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:STRack[:STATe]
value: bool = driver.calculate.marker.function.strack.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:STRack[:STATe]
driver.calculate.marker.function.strack.state.set(state = False, window =
↳recap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.15.2 Threshold**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNCTION:STRack:THReshold
--

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:STRack:THReshold value: float = driver.calculate.marker.function.strack.threshold.get(window = ↵ ↵repcap.Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

level: No help available

set(level: float, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:STRack:THReshold driver.calculate.marker.function.strack.threshold.set(level = 1.0, window = ↵ ↵repcap.Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param level

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.15.3 Trace

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:STRack:TRACe`

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:STRack:TRACe
value: float = driver.calculate.marker.function.strack.trace.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

trace_number: No help available

set(*trace_number: float, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:STRack:TRACe
driver.calculate.marker.function.strack.trace.set(trace_number = 1.0, window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param trace_number

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.16 Summary

class SummaryCls

Summary commands group definition. 20 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.clone()
```

Subgroups

6.2.4.3.16.1 Aoff

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:AOFF
driver.calculate.marker.function.summary.aoff.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.3.16.2 Average

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMmary:AVERage
value: bool = driver.calculate.marker.function.summary.average.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMmary:AVERage
driver.calculate.marker.function.summary.average.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.16.3 Mean

class MeanCls

Mean commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.mean.clone()
```


Subgroups

6.2.4.3.16.4 Average

class AverageCls

Average commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.mean.average.clone()
```

Subgroups

6.2.4.3.16.5 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMMary:MEAN:AVERage:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMMary:MEAN:AVERage:RESult
value: float = driver.calculate.marker.function.summary.mean.average.result.
↳ get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

mean_power: No help available

6.2.4.3.16.6 Phold

class PholdCls

Phold commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.mean.phold.clone()
```

Subgroups

6.2.4.3.16.7 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMMary:MEAN:PHOLd:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMMary:MEAN:PHOLd:RESult
value: float = driver.calculate.marker.function.summary.mean.phold.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

mean_power: No help available

6.2.4.3.16.8 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMMary:MEAN:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMMary:MEAN:RESult
value: float = driver.calculate.marker.function.summary.mean.result.get(window,
↳= repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

mean_power: No help available

6.2.4.3.16.9 State**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:MEAN:STATE
--

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:MEAN[:STATE] value: bool = driver.calculate.marker.function.summary.mean.state.get(window = ↳repcap.Window.Default, marker = repcap.Marker.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:MEAN[:STATE] driver.calculate.marker.function.summary.mean.state.set(state = False, window = ↳repcap.Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.16.10 Phold

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PHOLD`

class PholdCls

Phold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PHOLD
value: bool = driver.calculate.marker.function.summary.phold.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PHOLD
driver.calculate.marker.function.summary.phold.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.16.11 Ppeak

class PpeakCls

Ppeak commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.ppeak.clone()
```

Subgroups

6.2.4.3.16.12 Average

class AverageCls

Average commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.ppeak.average.clone()
```

Subgroups

6.2.4.3.16.13 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PPEak:AVERage:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PPEak:AVERage:RESult
value: float = driver.calculate.marker.function.summary.ppeak.average.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

```
return
    peak_power: No help available
```

6.2.4.3.16.14 Phold

class PholdCls

Phold commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.ppeak.phold.clone()
```

Subgroups

6.2.4.3.16.15 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:SUMMary:PPEak:PHOLd:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:SUMMary:PPEak:PHOLd:RESult
value: float = driver.calculate.marker.function.summary.ppeak.phold.result.
↳ get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

peak_power: No help available

6.2.4.3.16.16 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PPEak:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PPEak:RESult
value: float = driver.calculate.marker.function.summary.ppeak.result.get(window=
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

peak_power: No help available

6.2.4.3.16.17 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PPEak:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PPEak[:STATe]
value: bool = driver.calculate.marker.function.summary.ppeak.state.get(window =
↳ repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:PPEak[:STATE]
driver.calculate.marker.function.summary.ppeak.state.set(state = False, window_
↵= repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.16.18 Rms

class RmsCls

Rms commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.rms.clone()
```

Subgroups

6.2.4.3.16.19 Average

class AverageCls

Average commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.rms.average.clone()
```


Subgroups

6.2.4.3.16.20 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMmary:RMS:AVERage:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMmary:RMS:AVERage:RESult
value: float = driver.calculate.marker.function.summary.rms.average.result.
←get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

rms_power: No help available

6.2.4.3.16.21 Phold

class PholdCls

Phold commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.rms.phold.clone()
```

Subgroups

6.2.4.3.16.22 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMmary:RMS:PHOLd:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMmary:RMS:PHOLd:RESult
value: float = driver.calculate.marker.function.summary.rms.phold.result.
↳get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

rms_power: No help available

6.2.4.3.16.23 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:SUMmary:RMS:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:SUMmary:RMS:RESult
value: float = driver.calculate.marker.function.summary.rms.result.get(window =
↳repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

rms_power: No help available

6.2.4.3.16.24 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:RMS:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:RMS[:STATe]
value: bool = driver.calculate.marker.function.summary.rms.state.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:RMS[:STATe]
driver.calculate.marker.function.summary.rms.state.set(state = False, window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.16.25 StandardDev

class StandardDevCls

StandardDev commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.standardDev.clone()
```

Subgroups

6.2.4.3.16.26 Average

class AverageCls

Average commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.standardDev.average.clone()
```

Subgroups

6.2.4.3.16.27 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:AVERage:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:SDEVIation:AVERage:RESult
value: float = driver.calculate.marker.function.summary.standardDev.average.
    ↳ result.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return
standard_deviation: No help available

6.2.4.3.16.28 Phold

class PholdCls

Phold commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.summary.standardDev.phold.clone()
```

Subgroups

6.2.4.3.16.29 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:SUMMARY:SDEViation:PHOLd:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:SUMMARY:SDEViation:PHOLd:RESult
value: float = driver.calculate.marker.function.summary.standardDev.phold.
    ↪result.get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

standard_deviation: No help available

6.2.4.3.16.30 Result

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:RESult`

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default, marker=Marker.Default) → float`

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:SDEVIation:RESult
value: float = driver.calculate.marker.function.summary.standardDev.result.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

standard_deviation: No help available

6.2.4.3.16.31 State

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default, marker=Marker.Default) → bool`

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:SDEVIation[:STATe]
value: bool = driver.calculate.marker.function.summary.standardDev.state.
↪get(window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY:SDEVIation[:STATe]
driver.calculate.marker.function.summary.standardDev.state.set(state = False,
↪window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.3.16.32 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY[:STATe]
value: bool = driver.calculate.marker.function.summary.state.get(window =
↪repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCTION:SUMMARY[:STATe]
driver.calculate.marker.function.summary.state.set(state = False, window =
↪repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.3.17 Toi

class ToiCls

Toi commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.toi.clone()
```

Subgroups

6.2.4.3.17.1 Result

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNCtion:TOI:RESult
```

class ResultCls

Result commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNCtion:TOI:RESult
value: float = driver.calculate.marker.function.toi.result.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

toi: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.function.toi.result.clone()
```

Subgroups

6.2.4.3.17.2 Maximum

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:TOI:RESult:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:TOI:RESult:MAXimum
value: float = driver.calculate.marker.function.toi.result.maximum.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

toi: No help available

6.2.4.3.17.3 Minimum

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:FUNction:TOI:RESult:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:FUNction:TOI:RESult:MINimum
value: float = driver.calculate.marker.function.toi.result.minimum.get(window = ↵
↵repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

toi: No help available

6.2.4.3.17.4 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:FUNCTION:TOI:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:TOI[:STATE] value: bool = driver.calculate.marker.function.toi.state.get(window = repcap. ↳Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:FUNCTION:TOI[:STATE] driver.calculate.marker.function.toi.state.set(state = False, window = repcap. ↳Window.Default, marker = repcap.Marker.Default)</pre>
--

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.4 Link

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:LINK
value: bool = driver.calculate.marker.link.get(window = repcap.Window.Default,
↵marker = repcap.Marker.Default)
```

Links the specified marker in all displays of the specified type.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:LINK
driver.calculate.marker.link.set(state = False, window = repcap.Window.Default,
↵marker = repcap.Marker.Default)
```

Links the specified marker in all displays of the specified type.

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.5 LinkTo

class LinkToCls

LinkTo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.linkTo.clone()
```

Subgroups

6.2.4.5.1 Marker<MarkerDestination>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.calculate.marker.linkTo.marker.repcap_markerDestination_get()
driver.calculate.marker.linkTo.marker.repcap_markerDestination_set(repcap.
↳MarkerDestination.Nr1)
```

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LINK:TO:MARKer<MarkerDestination>
```

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: MarkerDestination, default value after init: MarkerDestination.Nr1

set(state: bool, window=Window.Default, marker=Marker.Default, markerDestination=MarkerDestination.Default) → None

```
# SCPI: CALCulate<n>:MARKer<ms>:LINK:TO:MARKer<mt>
driver.calculate.marker.linkTo.marker.set(state = False, window = repcap.Window.
↳Default, marker = repcap.Marker.Default, markerDestination = repcap.
↳MarkerDestination.Default)
```

This command links the normal source marker <ms> to any active destination marker <md> (normal or delta marker) . If you change the horizontal position of marker <md>, marker <ms> changes its horizontal position to the same value.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param markerDestination

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.linkTo.marker.clone()
```

6.2.4.6 LoExclude

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:LOEXclude
```

class LoExcludeCls

LoExclude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:LOEXclude
value: bool = driver.calculate.marker.loExclude.get(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:LOEXclude
driver.calculate.marker.loExclude.set(state = False, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.7 Maximum

class MaximumCls

Maximum commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.maximum.clone()
```

Subgroups

6.2.4.7.1 Auto

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:AUTO
driver.calculate.marker.maximum.auto.set(state = False, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.7.2 Left

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:LEFT
driver.calculate.marker.maximum.left.set(window = repcap.Window.Default, marker_
↳= repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the left of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.7.3 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:NEXT
driver.calculate.marker.maximum.next.set(window = repcap.Window.Default, marker_
↳= repcap.Marker.Default)
```

This command moves a marker to the next positive peak.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.7.4 Peak

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK`

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum[:PEAK]
driver.calculate.marker.maximum.peak.set(window = repcap.Window.Default, marker_
↳ marker = repcap.Marker.Default)
```

This command moves a marker to the highest level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.7.5 Right

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MAXimum:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MAXimum:RIGHT
driver.calculate.marker.maximum.right.set(window = repcap.Window.Default,
↳ marker = repcap.Marker.Default)
```

This command moves a marker to the next positive peak. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.8 Minimum**class MinimumCls**

Minimum commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.minimum.clone()
```

Subgroups**6.2.4.8.1 Auto****SCPI Commands**

```
CALCulate<Window>:MARKer<Marker>:MINimum:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:AUTO
driver.calculate.marker.minimum.auto.set(state = False, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.8.2 Left

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:LEFT`

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:LEFT
driver.calculate.marker.minimum.left.set(window = repcap.Window.Default, marker_
↳= repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.8.3 Next

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:MINimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:NEXT
driver.calculate.marker.minimum.next.set(window = repcap.Window.Default, marker_
↳= repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.8.4 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum[:PEAK]
driver.calculate.marker.minimum.peak.set(window = repcap.Window.Default, marker_
↳ marker = repcap.Marker.Default)
```

This command moves a marker to the minimum level. If the marker is not yet active, the command first activates the marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.8.5 Right

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:MINimum:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

```
# SCPI: CALCulate<n>:MARKer<m>:MINimum:RIGHT
driver.calculate.marker.minimum.right.set(window = repcap.Window.Default,
↳ marker = repcap.Marker.Default)
```

This command moves a marker to the next minimum peak value. The search includes only measurement values to the right of the current marker position.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.2.4.9 Pexcursion

SCPI Commands

CALCulate<Window>:MARKer<Marker>:PEXCursion

class PexcursionCls

Pexcursion commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:PEXCursion
value: float = driver.calculate.marker.pexcursion.get(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

excursion: No help available

set(excursion: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:PEXCursion
driver.calculate.marker.pexcursion.set(excursion = 1.0, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

This command defines the peak excursion (for all markers) . The peak excursion sets the requirements for a peak to be detected during a peak search. The unit depends on the measurement.

param excursion

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.10 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.clone()
```

Subgroups

6.2.4.10.1 Frame

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:FRAMe
```

class FrameCls

Frame commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:FRAMe
value: float = driver.calculate.marker.spectrogram.frame.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

frame: No help available

set(frame: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:FRAMe
driver.calculate.marker.spectrogram.frame.set(frame = 1.0, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param frame

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.10.2 Sarea**SCPI Commands**

`CALCulate<Window>:MARKer<Marker>:SPECTrogram:SARea`

class SareaCls

Sarea commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → SearchArea

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:SARea
value: enums.SearchArea = driver.calculate.marker.spectrogram.sarea.get(window_
↳= repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

search_area: No help available

set(*search_area: SearchArea, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:SARea
driver.calculate.marker.spectrogram.sarea.set(search_area = enums.SearchArea.
↳MEMory, window = repcap.Window.Default, marker = repcap.Marker.Default)
```

No command help available

param search_area

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.10.3 Xy

class XyCls

Xy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.xy.clone()
```

Subgroups

6.2.4.10.3.1 Maximum

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.xy.maximum.clone()
```

Subgroups

6.2.4.10.3.2 Peak

SCPI Commands

CALCulate<Window>:MARKer<Marker>:SPECTrogram:XY:MAXimum:PEAK

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:XY:MAXimum[:PEAK]
driver.calculate.marker.spectrogram.xy.maximum.peak.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.3.3 Minimum

class MinimumCls

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.xy.minimum.clone()
```

Subgroups

6.2.4.10.3.4 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECtrogram:XY:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECtrogram:XY:MINimum[:PEAK]
driver.calculate.marker.spectrogram.xy.minimum.peak.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4 Y

class YCls

Y commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.y.clone()
```

Subgroups

6.2.4.10.4.1 Maximum

class MaximumCls

Maximum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.y.maximum.clone()
```

Subgroups

6.2.4.10.4.2 Above

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:ABOVE
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MAXimum:ABOVE
driver.calculate.marker.spectrogram.y.maximum.above.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.3 Below

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:SPEctrogram:Y:MAXimum:BELOW`

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:Y:MAXimum:BELOW
driver.calculate.marker.spectrogram.y.maximum.below.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.4 Next

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:SPEctrogram:Y:MAXimum:NEXT`

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:Y:MAXimum:NEXT
driver.calculate.marker.spectrogram.y.maximum.next.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.5 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPEctrogram:Y:MAXimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPEctrogram:Y:MAXimum[:PEAK]
driver.calculate.marker.spectrogram.y.maximum.peak.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.6 Minimum

class MinimumCls

Minimum commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.spectrogram.y.minimum.clone()
```

Subgroups

6.2.4.10.4.7 Above

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPEctrogram:Y:MINimum:ABOVE
```

class AboveCls

Above commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:ABOVE
driver.calculate.marker.spectrogram.y.minimum.above.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.8 Below

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:BELOW
```

class BelowCls

Below commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:BELOW
driver.calculate.marker.spectrogram.y.minimum.below.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.9 Next

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum:NEXT
driver.calculate.marker.spectrogram.y.minimum.next.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.10.4.10 Peak

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MINimum:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:SPECTrogram:Y:MINimum[:PEAK]
driver.calculate.marker.spectrogram.y.minimum.peak.set(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

set_with_opc(window=Window.Default, marker=Marker.Default, opc_timeout_ms: int = -1) → None

6.2.4.11 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
value: bool = driver.calculate.marker.state.get(window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>[:STATe]
driver.calculate.marker.state.set(state = False, window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

This command turns markers on and off. If the corresponding marker number is currently active as a delta marker, it is turned into a normal marker.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.12 Trace

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:TRACe
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
value: float = driver.calculate.marker.trace.get(window = repcap.Window.Default,
↪ marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

trace: No help available

set(trace: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:TRACe
driver.calculate.marker.trace.set(trace = 1.0, window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

This command selects the trace the marker is positioned on. Note that the corresponding trace must have a trace mode other than 'Blank'. If necessary, the command activates the marker first.

param trace

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.13 X

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X
```

class XCls

X commands group definition. 6 total commands, 2 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X
value: float = driver.calculate.marker.x.get(window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

stimulus: No help available

set(stimulus: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X
driver.calculate.marker.x.set(stimulus = 1.0, window = repcap.Window.Default,
↪marker = repcap.Marker.Default)
```

This command moves a marker to a specific coordinate on the x-axis. If necessary, the command activates the marker. If the marker has been used as a delta marker, the command turns it into a normal marker.

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.x.clone()
```

Subgroups

6.2.4.13.1 Slimits

class SlimitsCls

Slimits commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.x.slimits.clone()
```


Subgroups

6.2.4.13.1.1 Left

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X:SLIMits:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:LEFT
value: float = driver.calculate.marker.x.slimits.left.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

search_limit: No help available

set(search_limit: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:LEFT
driver.calculate.marker.x.slimits.left.set(search_limit = 1.0, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param search_limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.13.1.2 Right

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:X:SLIMits:RIGHT`

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → float

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:RIGHT
value: float = driver.calculate.marker.x.slimits.right.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

limit: No help available

set(*limit: float, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:RIGHT
driver.calculate.marker.x.slimits.right.set(limit = 1.0, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param limit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

6.2.4.13.1.3 State

SCPI Commands

CALCulate<Window>:MARKer<Marker>:X:SLIMits:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, marker=Marker.Default*) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits[:STATe]
value: bool = driver.calculate.marker.x.slimits.state.get(window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

state: No help available

set(*state: bool, window=Window.Default, marker=Marker.Default*) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits[:STATe]
driver.calculate.marker.x.slimits.state.set(state = False, window = repcap.
↳Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.13.1.4 Zoom

class ZoomCls

Zoom commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.x.slimits.zoom.clone()
```

Subgroups

6.2.4.13.1.5 State

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:X:SLIMits:ZOOM:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → bool

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:ZOOM[:STATe]
value: bool = driver.calculate.marker.x.slimits.zoom.state.get(window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

state: No help available

set(state: bool, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:X:SLIMits:ZOOM[:STATe]
driver.calculate.marker.x.slimits.zoom.state.set(state = False, window = repcap.
↳ Window.Default, marker = repcap.Marker.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.13.2 Ssize**SCPI Commands**

CALCulate<Window>:MARKer<Marker>:X:SSIZE
--

class SsizeCls

Ssize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → Stepsize

<pre># SCPI: CALCulate<n>:MARKer<m>:X:SSIZE value: enums.Stepsize = driver.calculate.marker.x.ssize.get(window = repcap. ↳Window.Default, marker = repcap.Marker.Default)</pre>

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

stepsize: No help available

set(stepsize: Stepsize, window=Window.Default, marker=Marker.Default) → None

<pre># SCPI: CALCulate<n>:MARKer<m>:X:SSIZE driver.calculate.marker.x.ssize.set(stepsize = enums.Stepsize.POINTs, window = ↳repcap.Window.Default, marker = repcap.Marker.Default)</pre>
--

No command help available

param stepsize

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

6.2.4.14 Y

SCPI Commands

`CALCulate<Window>:MARKer<Marker>:Y`

class YCls

Y commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y
value: float = driver.calculate.marker.y.get(window = repcap.Window.Default,
↵marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

return

value_stimulus: No help available

set(value_stimulus: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:Y
driver.calculate.marker.y.set(value_stimulus = 1.0, window = repcap.Window.
↵Default, marker = repcap.Marker.Default)
```

Queries the result at the position of the specified marker.

param value_stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Marker’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.marker.y.clone()
```

Subgroups

6.2.4.14.1 Percent

SCPI Commands

```
CALCulate<Window>:MARKer<Marker>:Y:PERCent
```

class PercentCls

Percent commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, marker=Marker.Default) → float

```
# SCPI: CALCulate<n>:MARKer<m>:Y:PERCent
value: float = driver.calculate.marker.y.percent.get(window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

return

probability: No help available

set(probability: float, window=Window.Default, marker=Marker.Default) → None

```
# SCPI: CALCulate<n>:MARKer<m>:Y:PERCent
driver.calculate.marker.y.percent.set(probability = 1.0, window = repcap.Window.
↳Default, marker = repcap.Marker.Default)
```

No command help available

param probability

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param marker

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Marker')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.math.expression.clone()
```

Subgroups

6.2.5.1.1 Define

SCPI Commands

```
CALCulate<Window>:MATH:EXPRession:DEFine
```

class DefineCls

Define commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: CALCulate<n>:MATH[:EXPRession][:DEFine]
value: str = driver.calculate.math.expression.define.get(window = repcap.Window.
↳Default)
```

This command selects the operation for trace mathematics.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

expression: No help available

set(expression: str, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MATH[:EXPRession][:DEFine]
driver.calculate.math.expression.define.set(expression = r1, window = repcap.
↳Window.Default)
```

This command selects the operation for trace mathematics.

param expression

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.5.2 Mode

SCPI Commands

`CALCulate<Window>:MATH:MODE`

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AverageModeA

```
# SCPI: CALCulate<n>:MATH:MODE
value: enums.AverageModeA = driver.calculate.math.mode.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

mode: No help available

set(mode: AverageModeA, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MATH:MODE
driver.calculate.math.mode.set(mode = enums.AverageModeA.LINear, window =
↳repcap.Window.Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.5.3 Position

SCPI Commands

`CALCulate<Window>:MATH:POSition`

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:MATH:POSition
value: float = driver.calculate.math.position.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

position: No help available

set(*position: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:MATH:POSition
driver.calculate.math.position.set(position = 1.0, window = repcap.Window.
↪Default)
```

No command help available

param position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.5.4 State

SCPI Commands

```
CALCulate<Window>:MATH:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:MATH:STATe
value: bool = driver.calculate.math.state.get(window = repcap.Window.Default)
```

This command turns trace mathematics on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: ON | 1 Turns trace mathematics on and selects the operation that has been selected last (or (TRACE1-TRACE3) if you have not yet selected one) . OFF | 0 Turns trace mathematics off.

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:MATH:STATe
driver.calculate.math.state.set(state = False, window = repcap.Window.Default)
```

This command turns trace mathematics on and off.

param state

ON | 1 Turns trace mathematics on and selects the operation that has been selected

last (or (TRACE1-TRACE3) if you have not yet selected one) . OFF | 0 Turns trace mathematics off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.6 Msra

class MsraCls

Msra commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.msra.clone()
```

Subgroups

6.2.6.1 Aline

class AlineCls

Aline commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.msra.aline.clone()
```

Subgroups

6.2.6.1.1 Show

SCPI Commands

```
CALCulate<Window>:MSRA:ALIne:SHOW
```

class ShowCls

Show commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:MSRA:ALIne:SHOW
value: bool = driver.calculate.msra.aline.show.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MSRA:ALINe:SHOW
driver.calculate.msra.aline.show.set(state = False, window = repcap.Window.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.6.1.2 Value

SCPI Commands

```
CALCulate<Window>:MSRA:ALINe:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:MSRA:ALINe:VALue
value: float = driver.calculate.msra.aline.value.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

stimulus: No help available

set(stimulus: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:MSRA:ALINe:VALue
driver.calculate.msra.aline.value.set(stimulus = 1.0, window = repcap.Window.
↳Default)
```

No command help available

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.6.2 Window

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.msra.window.clone()
```

Subgroups

6.2.6.2.1 Ival

SCPI Commands

```
CALCulate<Window>:MSRA:WINDow:IVAL
```

class IvalCls

Ival commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class TimeSpan

Response structure. Fields:

- Start: float: No parameter help available
- Stop: float: No parameter help available

get(stimulus: float, window=Window.Default) → TimeSpan

```
# SCPI: CALCulate<n>:MSRA:WINDow:IVAL
value: TimeSpan = driver.calculate.msra.window.ival.get(stimulus = 1.0, window_
↪= repcap.Window.Default)
```

No command help available

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

structure: for return value, see the help for TimeSpan structure arguments.

6.2.7 PeakSearch

class PeakSearchCls

PeakSearch commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.peakSearch.clone()
```

Subgroups

6.2.7.1 Auto

SCPI Commands

```
CALCulate<Window>:PSEarch:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:PSEarch:AUTO
driver.calculate.peakSearch.auto.set(state = False, window = repcap.Window.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.7.2 Margin

SCPI Commands

```
CALCulate<Window>:PSEarch:MARGin
```

class MarginCls

Margin commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:PSEarch:MARGin
value: float = driver.calculate.peakSearch.margin.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

threshold: No help available

set(*threshold: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:PSEarch:MARGin
driver.calculate.peakSearch.margin.set(threshold = 1.0, window = repcap.Window.
↪Default)
```

No command help available

param threshold

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.7.3 Pshow

SCPI Commands

```
CALCulate<Window>:PSEarch:PSHow
```

class PshowCls

Pshow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:PSEarch:PSHow
value: bool = driver.calculate.peakSearch.pshow.get(window = repcap.Window.
↪Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:PSEarch:PSHow
driver.calculate.peakSearch.pshow.set(state = False, window = repcap.Window.
↪Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.7.4 Subranges**SCPI Commands**

CALCulate<Window>:PSEarch:SUBRanges

class SubrangesCls

Subranges commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

<pre># SCPI: CALCulate<n>:PSEarch:SUBRanges value: float = driver.calculate.peakSearch.subranges.get(window = repcap.Window.Default)</pre>
--

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

number_peaks: No help available

set(number_peaks: float, window=Window.Default) → None

<pre># SCPI: CALCulate<n>:PSEarch:SUBRanges driver.calculate.peakSearch.subranges.set(number_peaks = 1.0, window = repcap.Window.Default)</pre>

No command help available

param number_peaks

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.8 Pmeter<PowerMeter>**RepCap Settings**

<pre># Range: Nr1 .. Nr16 rc = driver.calculate.pmeter.repcap_powerMeter_get() driver.calculate.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)</pre>

class PmeterCls

Pmeter commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.pmeter.clone()
```

Subgroups

6.2.8.1 Relative

class RelativeCls

Relative commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.pmeter.relative.clone()
```

Subgroups

6.2.8.1.1 Magnitude

SCPI Commands

```
CALCulate<Window>:PMETer<PowerMeter>:RELative:MAGNitude
```

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, powerMeter=PowerMeter.Default) → float

```
# SCPI: CALCulate<n>:PMETer<p>:RELative[:MAGNitude]
value: float = driver.calculate.pmeter.relative.magnitude.get(window = repcap.
↳Window.Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

ref_value: No help available

set(ref_value: float, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALCulate<n>:PMETer<p>:RELative[:MAGNitude]
driver.calculate.pmeter.relative.magnitude.set(ref_value = 1.0, window = repcap.
↳ Window.Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param ref_value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.2.8.1.2 State

SCPI Commands

```
CALCulate<Window>:PMETer<PowerMeter>:RELative:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, powerMeter=PowerMeter.Default) → bool

```
# SCPI: CALCulate<n>:PMETer<p>:RELative:STATE
value: bool = driver.calculate.pmeter.relative.state.get(window = repcap.Window.
↳ Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(state: bool, window=Window.Default, powerMeter=PowerMeter.Default) → None

```
# SCPI: CALCulate<n>:PMETer<p>:RELative:STATE
driver.calculate.pmeter.relative.state.set(state = False, window = repcap.
↳ Window.Default, powerMeter = repcap.PowerMeter.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Pmeter’)

6.2.9 Rtms

class RtmsCls

Rtms commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.rtms.clone()
```

Subgroups

6.2.9.1 Aline

class AlineCls

Aline commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.rtms.aline.clone()
```

Subgroups

6.2.9.1.1 Show

SCPI Commands

```
CALCulate<Window>:RTMS:ALIne:SHOW
```

class ShowCls

Show commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:RTMS:ALIne:SHOW
value: bool = driver.calculate.rtms.aline.show.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:RTMS:ALINe:SHOW
driver.calculate.rtms.aline.show.set(state = False, window = repcap.Window.
↪Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.9.1.2 Value

SCPI Commands

CALCulate<Window>:RTMS:ALINe:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:RTMS:ALINe:VALue
value: float = driver.calculate.rtms.aline.value.get(window = repcap.Window.
↪Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

stimulus: No help available

set(stimulus: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:RTMS:ALINe:VALue
driver.calculate.rtms.aline.value.set(stimulus = 1.0, window = repcap.Window.
↪Default)
```

No command help available

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.9.2 Window

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.rtms.window.clone()
```

Subgroups

6.2.9.2.1 Ival

SCPI Commands

```
CALCulate<Window>:RTMS:WINDow:IVAL
```

class IvalCls

Ival commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class TimeSpan

Response structure. Fields:

- Start: float: No parameter help available
- Stop: float: No parameter help available

get(stimulus: float, window=Window.Default) → TimeSpan

```
# SCPI: CALCulate<n>:RTMS:WINDow:IVAL
value: TimeSpan = driver.calculate.rtms.window.ival.get(stimulus = 1.0, window_
↪= repcap.Window.Default)
```

No command help available

param stimulus

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

structure: for return value, see the help for TimeSpan structure arguments.

6.2.10 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 9 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrogram.clone()
```

Subgroups

6.2.10.1 Clear

class ClearCls

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrogram.clear.clone()
```

Subgroups

6.2.10.1.1 Immediate

SCPI Commands

```
CALCulate<Window>:SPECtrogram:CLEar:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram:CLEar[:IMMediate]
driver.calculate.spectrogram.clear.immediate.set(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

set_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

6.2.10.2 Continuous

SCPI Commands

`CALCulate<Window>:SPECtrogram:CONTinuous`

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:SPECtrogram:CONTinuous
value: bool = driver.calculate.spectrogram.continuous.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram:CONTinuous
driver.calculate.spectrogram.continuous.set(state = False, window = repcap.
↳Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.10.3 Frame

class FrameCls

Frame commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrogram.frame.clone()
```


Subgroups

6.2.10.3.1 Count

SCPI Commands

```
CALCulate<Window>:SPECtrogram:FRAMe:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:SPECtrogram:FRAMe:COUNT
value: float = driver.calculate.spectrogram.frame.count.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

frames: No help available

set(*frames: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECtrogram:FRAMe:COUNT
driver.calculate.spectrogram.frame.count.set(frames = 1.0, window = repcap.
↳Window.Default)
```

No command help available

param frames

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.10.3.2 Select

SCPI Commands

```
CALCulate<Window>:SPECtrogram:FRAMe:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*frame: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECtrogram:FRAMe:SElect
driver.calculate.spectrogram.frame.select.set(frame = 1.0, window = repcap.
↳Window.Default)
```

No command help available

param frame

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.10.4 Hdepth

SCPI Commands

CALCulate<Window>:SPECtrogram:HDEPth

class HdepthCls

Hdepth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:SPECtrogram:HDEPth
value: float = driver.calculate.spectrogram.hdepth.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

history: No help available

set(*history: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECtrogram:HDEPth
driver.calculate.spectrogram.hdepth.set(history = 1.0, window = repcap.Window.
↳Default)
```

No command help available

param history

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.10.5 State

SCPI Commands

CALCulate<Window>:SPECtrogram:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:SPECtrogram[:STATe]
value: bool = driver.calculate.spectrogram.state.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:SPECtrogram[:STATe]
driver.calculate.spectrogram.state.set(state = False, window = repcap.Window.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.10.6 ThreeDim

class ThreeDimCls

ThreeDim commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrogram.threeDim.clone()
```

Subgroups

6.2.10.6.1 State

SCPI Commands

`CALCulate<Window>:SPECTrogram:THReedim:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:SPECTrogram:THReedim[:STATe]
value: bool = driver.calculate.spectrogram.threeDim.state.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECTrogram:THReedim[:STATe]
driver.calculate.spectrogram.threeDim.state.set(state = False, window = repcap.
↳Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.10.7 Tstamp

class TstampCls

Tstamp commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.spectrogram.tstamp.clone()
```

Subgroups

6.2.10.7.1 Data

SCPI Commands

```
CALCulate<Window>:SPECtrogram:TSTamp:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Seconds: float: No parameter help available
- Nanoseconds: float: No parameter help available
- Reserved: float: No parameter help available
- Reserved_B: float: No parameter help available

get(frames: SelectionRangeB, window=Window.Default) → GetStruct

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp:DATA
value: GetStruct = driver.calculate.spectrogram.tstamp.data.get(frames = enums.
↳ SelectionRangeB.ALL, window = repcap.Window.Default)
```

No command help available

param frames

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

structure: for return value, see the help for GetStruct structure arguments.

6.2.10.7.2 State

SCPI Commands

```
CALCulate<Window>:SPECtrogram:TSTamp:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → bool

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp[:STATe]
value: bool = driver.calculate.spectrogram.tstamp.state.get(window = repcap.
↳ Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(*state: bool, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:SPECtrogram:TSTamp[:STATe]
driver.calculate.spectrogram.tstamp.state.set(state = False, window = repcap.
↳ Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.11 Statistics

SCPI Commands

CALCulate<Window>:STATistics:PRESet

class StatisticsCls

Statistics commands group definition. 11 total commands, 5 Subgroups, 1 group commands

preset(*window=Window.Default*) → None

```
# SCPI: CALCulate<n>:STATistics:PRESet
driver.calculate.statistics.preset(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

preset_with_opc(*window=Window.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.clone()
```

Subgroups

6.2.11.1 AmplitudeProbDensity

class AmplitudeProbDensityCls

AmplitudeProbDensity commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.amplitudeProbDensity.clone()
```

Subgroups

6.2.11.1.1 State

SCPI Commands

```
CALCulate<Window>:STATistics:APD:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:STATistics:APD[:STATe]
value: bool = driver.calculate.statistics.amplitudeProbDensity.state.get(window,
↳ repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:APD[:STATe]
driver.calculate.statistics.amplitudeProbDensity.state.set(state = False,
↳ window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.11.2 CumulativeDistribFnc

class CumulativeDistribFncCls

CumulativeDistribFnc commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.cumulativeDistribFnc.clone()
```

Subgroups

6.2.11.2.1 State

SCPI Commands

```
CALCulate<Window>:STATistics:CCDF:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:STATistics:CCDF[:STATe]
value: bool = driver.calculate.statistics.cumulativeDistribFnc.state.get(window,
↳= repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:CCDF[:STATe]
driver.calculate.statistics.cumulativeDistribFnc.state.set(state = False,
↳window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.11.2.2 X<Trace>**RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.calculate.statistics.cumulativeDistribFnc.x.repcap_trace_get()
driver.calculate.statistics.cumulativeDistribFnc.x.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:STATistics:CCDF:X<Trace>
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(probability: Probability, window=Window.Default, trace=Trace.Default) → float

```
# SCPI: CALCulate<n>:STATistics:CCDF:X<t>
value: float = driver.calculate.statistics.cumulativeDistribFnc.x.
    ↪get(probability = enums.Probability.P0_01, window = repcap.Window.Default,
    ↪trace = repcap.Trace.Default)
```

No command help available

param probability

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘X’)

return

ccdf_result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.cumulativeDistribFnc.x.clone()
```

6.2.11.3 Nsamples

SCPI Commands

`CALCulate<Window>:STATistics:NSAMples`

class NsamplesCls

Nsamples commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:STATistics:NSAMples
value: float = driver.calculate.statistics.nsamples.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

samples: No help available

set(*samples: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:STATistics:NSAMples
driver.calculate.statistics.nsamples.set(samples = 1.0, window = repcap.Window.
↳Default)
```

No command help available

param samples

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.11.4 Result<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.calculate.statistics.result.repcap_trace_get()
driver.calculate.statistics.result.repcap_trace_set(repcap.Trace.Tr1)
```

SCPI Commands

```
CALCulate<Window>:STATistics:RESult<Trace>
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

get(*result_type*: ResultTypeB, *window*=Window.Default, *trace*=Trace.Default) → List[float]

```
# SCPI: CALCulate<n>:STATistics:RESult<res>
value: List[float] = driver.calculate.statistics.result.get(result_type = enums.
    ↳ResultTypeB.ALL, window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param result_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Result’)

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.result.clone()
```

6.2.11.5 Scale

class ScaleCls

Scale commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.scale.clone()
```

Subgroups

6.2.11.5.1 X

class XCls

X commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.scale.x.clone()
```

Subgroups

6.2.11.5.1.1 Range

SCPI Commands

```
CALCulate<Window>:STATistics:SCALE:X:RANGe
```

class RangeCls

Range commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:X:RANGe
value: float = driver.calculate.statistics.scale.x.range.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

range_py: No help available

set(range_py: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:X:RANGe
driver.calculate.statistics.scale.x.range.set(range_py = 1.0, window = repcap.
↳Window.Default)
```

No command help available

param range_py

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.11.5.1.2 RefLevel

SCPI Commands

```
CALCulate<Window>:STATistics:SCALE:X:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:X:RLEVel
value: float = driver.calculate.statistics.scale.x.refLevel.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

ref_level: No help available

set(ref_level: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:X:RLEVel
driver.calculate.statistics.scale.x.refLevel.set(ref_level = 1.0, window =
↳repcap.Window.Default)
```

No command help available

param ref_level

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.11.5.2 Y

class YCls

Y commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.statistics.scale.y.clone()
```

Subgroups

6.2.11.5.2.1 Lower

SCPI Commands

```
CALCulate<Window>:STATistics:SCALE:Y:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:LOWer
value: float = driver.calculate.statistics.scale.y.lower.get(window = repcap.
↳Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

magnitude: No help available

set(*magnitude: float, window=Window.Default*) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:LOWer
driver.calculate.statistics.scale.y.lower.set(magnitude = 1.0, window = repcap.
↳Window.Default)
```

No command help available

param magnitude

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.11.5.2.2 Unit

SCPI Commands

```
CALCulate<Window>:STATistics:SCALE:Y:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → ScaleYaxisUnit

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UNIT
value: enums.ScaleYaxisUnit = driver.calculate.statistics.scale.y.unit.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: No help available

set(unit: ScaleYaxisUnit, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UNIT
driver.calculate.statistics.scale.y.unit.set(unit = enums.ScaleYaxisUnit.ABS,
↪window = repcap.Window.Default)
```

No command help available

param unit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.11.5.2.3 Upper

SCPI Commands

CALCulate<Window>:STATistics:SCALE:Y:UPPer

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UPPer
value: float = driver.calculate.statistics.scale.y.upper.get(window = repcap.
↪Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

magnitude: No help available

set(magnitude: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:STATistics:SCALE:Y:UPPer
driver.calculate.statistics.scale.y.upper.set(magnitude = 1.0, window = repcap.
↪Window.Default)
```

No command help available

param magnitude
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.12 Threshold

SCPI Commands

CALCulate<Window>:THReshold

class ThresholdCls

Threshold commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: CALCulate<n>:THReshold
value: float = driver.calculate.threshold.get(window = repcap.Window.Default)
```

No command help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return
level: No help available

set(level: float, window=Window.Default) → None

```
# SCPI: CALCulate<n>:THReshold
driver.calculate.threshold.set(level = 1.0, window = repcap.Window.Default)
```

No command help available

param level
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.threshold.clone()
```


Subgroups

6.2.12.1 State

SCPI Commands

CALCulate<Window>:THReshold:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: CALCulate<n>:THReshold:STATe
value: bool = driver.calculate.threshold.state.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: CALCulate<n>:THReshold:STATe
driver.calculate.threshold.state.set(state = False, window = repcap.Window.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Calculate')

6.2.13 Unit

class UnitCls

Unit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calculate.unit.clone()
```

Subgroups

6.2.13.1 Angle

SCPI Commands

```
CALCulate<Window>:UNIT:ANGLE
```

class AngleCls

Angle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AngleUnit

```
# SCPI: CALCulate<n>:UNIT:ANGLE
value: enums.AngleUnit = driver.calculate.unit.angle.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: No help available

set(unit: AngleUnit, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNIT:ANGLE
driver.calculate.unit.angle.set(unit = enums.AngleUnit.DEG, window = repcap.
↳Window.Default)
```

No command help available

param unit

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.2.13.2 Power

SCPI Commands

CALCulate<Window>:UNIT:POWer

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → PowerUnitB

```
# SCPI: CALCulate<n>:UNIT:POWer
value: enums.PowerUnitB = driver.calculate.unit.power.get(window = repcap.
↳Window.Default)
```

This command selects the unit of the y-axis. The unit applies to all power-based measurement windows with absolute values.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

return

unit: DBM | V | A | W | DBPW | WATT | DBUV | DBMV | VOLT | DBUA | AMPere

set(unit: PowerUnitB, window=Window.Default) → None

```
# SCPI: CALCulate<n>:UNIT:POWer
driver.calculate.unit.power.set(unit = enums.PowerUnitB.A, window = repcap.
↳Window.Default)
```

This command selects the unit of the y-axis. The unit applies to all power-based measurement windows with absolute values.

param unit

DBM | V | A | W | DBPW | WATT | DBUV | DBMV | VOLT | DBUA | AMPere

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Calculate’)

6.3 Calibration

class CalibrationCls

Calibration commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calibration.clone()
```

Subgroups

6.3.1 Aiq

class AiqCls

Aiq commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calibration.aiq.clone()
```

Subgroups

6.3.1.1 HaTiming

class HaTimingCls

HaTiming commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calibration.aiq.haTiming.clone()
```

Subgroups

6.3.1.1.1 State

SCPI Commands

```
CALibration:AIQ:HaTiming:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: CALibration:AIQ:HaTiming[:STATE]
value: bool = driver.calibration.aiq.haTiming.state.get()
```

No command help available

return
 auto_mode: No help available

set(auto_mode: bool) → None

```
# SCPI: CALibration:AIQ:HaTiming[:STaTe]
driver.calibration.aiq.haTiming.state.set(auto_mode = False)
```

No command help available

param auto_mode
 No help available

6.3.2 All

SCPI Commands

CALibration:ALL

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(cal_state: Optional[CalibrationScope] = None) → None

```
# SCPI: CALibration[:ALL]
driver.calibration.all.set(cal_state = enums.CalibrationScope.AClear)
```

This command initiates a calibration (self-alignment) routine and queries if calibration was successful. During the acquisition of correction data the instrument does not accept any remote control commands. Note:If you start a self-alignment remotely, then select the ‘Local’ softkey while the alignment is still running, the instrument only returns to the manual operation state after the alignment is completed. In order to recognize when the acquisition of correction data is completed, the MAV bit in the status byte can be used. If the associated bit is set in the Service Request Enable (SRE) register, the instrument generates a service request after the acquisition of correction data has been completed.

param cal_state
 No help available

6.3.3 Due

class DueCls

Due commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.calibration.due.clone()
```

Subgroups

6.3.3.1 Days

SCPI Commands

```
CALibration:DUE:DAYS
```

class DaysCls

Days commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DaysStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Day_1: enums.DaysOfWeek: ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_2: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_3: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_4: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_5: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_6: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay
- Day_7: enums.DaysOfWeek: Optional setting parameter. ALL | MONDay | TUESday | WEDNesday | THURsday | FRIDay | SATurday | SUNDay

get() → DaysStruct

```
# SCPI: CALibration:DUE:DAYS
value: DaysStruct = driver.calibration.due.days.get()
```

Defines the days on which a self-alignment is scheduled for method RsFswp.Calibration.Due.Schedule.set ON. Up to 7 different days can be scheduled.

return

structure: for return value, see the help for DaysStruct structure arguments.

set(structure: DaysStruct) → None

```
# SCPI: CALibration:DUE:DAYS
structure = driver.calibration.due.days.DaysStruct()
structure.Day_1: enums.DaysOfWeek = enums.DaysOfWeek.ALL
```

(continues on next page)

(continued from previous page)

```

structure.Day_2: enums.DaysOfWeek = enums.DaysOfWeek.ALL
structure.Day_3: enums.DaysOfWeek = enums.DaysOfWeek.ALL
structure.Day_4: enums.DaysOfWeek = enums.DaysOfWeek.ALL
structure.Day_5: enums.DaysOfWeek = enums.DaysOfWeek.ALL
structure.Day_6: enums.DaysOfWeek = enums.DaysOfWeek.ALL
structure.Day_7: enums.DaysOfWeek = enums.DaysOfWeek.ALL
driver.calibration.due.days.set(structure)

```

Defines the days on which a self-alignment is scheduled for method RsFswp.Calibration.Due.Schedule.set ON. Up to 7 different days can be scheduled.

param structure

for set value, see the help for DaysStruct structure arguments.

6.3.3.2 Schedule

SCPI Commands

CALibration:DUE:SCchedule

class ScheduleCls

Schedule commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```

# SCPI: CALibration:DUE:SCchedule
value: bool = driver.calibration.due.schedule.get()

```

If enabled, a self-alignment is performed regularly at specific days and time. Specify the date and time using the method RsFswp.Calibration.Due.Days.set and method RsFswp.Calibration.Due.Time.set commands.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```

# SCPI: CALibration:DUE:SCchedule
driver.calibration.due.schedule.set(state = False)

```

If enabled, a self-alignment is performed regularly at specific days and time. Specify the date and time using the method RsFswp.Calibration.Due.Days.set and method RsFswp.Calibration.Due.Time.set commands.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.3.3.3 Shutdown

SCPI Commands

CALibration:DUE:SHUTdown

class ShutdownCls

Shutdown commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: CALibration:DUE:SHUTdown
driver.calibration.due.shutdown.set(state = False)
```

If activated, the R&S FSWP is automatically shut down after self-alignment is completed. Note that the instrument cannot be restarted via remote control.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.3.3.4 Time

SCPI Commands

CALibration:DUE:TIME

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: CALibration:DUE:TIME
value: str = driver.calibration.due.time.get()
```

Defines the time at which a self-alignment is scheduled for the days specified by method RsFswp.Calibration.Due.Days.set, if method RsFswp.Calibration.Due.Schedule.set ON.

return

time: string with format 'hh:mm' (24 hours)

set(time: str) → None

```
# SCPI: CALibration:DUE:TIME
driver.calibration.due.time.set(time = '1')
```

Defines the time at which a self-alignment is scheduled for the days specified by method RsFswp.Calibration.Due.Days.set, if method RsFswp.Calibration.Due.Schedule.set ON.

param time

string with format 'hh:mm' (24 hours)

6.3.3.5 Warmup

SCPI Commands

CALibration:DUE:WARMup

class WarmupCls

Warmup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: CALibration:DUE:WARMup
value: bool = driver.calibration.due.warmup.get()
```

If enabled, self-alignment is started automatically after the warmup operation has completed.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: CALibration:DUE:WARMup
driver.calibration.due.warmup.set(state = False)
```

If enabled, self-alignment is started automatically after the warmup operation has completed.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.3.4 PreSelection

SCPI Commands

CALibration:PRESelection

class PreSelectionCls

PreSelection commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: CALibration:PRESelection
driver.calibration.preSelection.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: CALibration:PRESelection
driver.calibration.preSelection.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param `opc_timeout_ms`

Maximum time to wait in milliseconds, valid only for this call.

6.3.5 Result

SCPI Commands

CALibration:REStult

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: CALibration:REStult
value: str = driver.calibration.result.get()
```

This command returns the results collected during calibration.

return

calibration_data: String containing the calibration data.

6.4 Configure

class ConfigureCls

Configure commands group definition. 17 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.clone()
```

Subgroups

6.4.1 Ademod

class AdemodCls

Ademod commands group definition. 13 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.clone()
```

Subgroups

6.4.1.1 Results

class ResultsCls

Results commands group definition. 13 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.clone()
```

Subgroups

6.4.1.1.1 Am

class AmCls

Am commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.am.clone()
```

Subgroups

6.4.1.1.1.1 Detector<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.configure.ademod.results.am.detector.repcap_trace_get()
driver.configure.ademod.results.am.detector.repcap_trace_set(repcap.Trace.Tr1)
```

class DetectorCls

Detector commands group definition. 4 total commands, 3 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.am.detector.clone()
```

Subgroups

6.4.1.1.1.2 Mode

SCPI Commands

```
CONFIgure:ADEMod:RESults:AM:DETEctor<Trace>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → TraceModeE

```
# SCPI: CONFIgure:ADEMod:RESults:AM:DETEctor<det>:MODE
value: enums.TraceModeE = driver.configure.ademod.results.am.detector.mode.
↪get(trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Detector’)

return

mode: WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERAGE The average result is determined over all sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

set(mode: TraceModeE, trace=Trace.Default) → None

```
# SCPI: CONFIgure:ADEMod:RESults:AM:DETEctor<det>:MODE
driver.configure.ademod.results.am.detector.mode.set(mode = enums.TraceModeE.
↪AVERAGE, trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param mode

WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERAGE The average result is determined over all sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Detector’)

6.4.1.1.1.3 Reference

SCPI Commands

```
CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:REFerence
```

class ReferenceCls

Reference commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: CONFigure:ADEMod:RESults:AM:DETEctor<det>:REFerence
value: float = driver.configure.ademod.results.am.detector.reference.get(trace_
↳= repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

ref_value: double value The unit depends on the demodulation type: ACV: V AM: % FM: Hz PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

set(*ref_value: float, trace=Trace.Default*) → None

```
# SCPI: CONFigure:ADEMod:RESults:AM:DETEctor<det>:REFerence
driver.configure.ademod.results.am.detector.reference.set(ref_value = 1.0,
↳trace = repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param ref_value

double value The unit depends on the demodulation type: ACV: V AM: % FM: Hz PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.am.detector.reference.clone()
```

Subgroups

6.4.1.1.1.4 MeastoRef<RefMeasurement>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.ademod.results.am.detector.reference.meastoRef.repcap_
↳refMeasurement_get()
driver.configure.ademod.results.am.detector.reference.meastoRef.repcap_refMeasurement_
↳set(repcap.RefMeasurement.Nr1)
```

SCPI Commands

```
CONFIGure:ADEMod:RESults:AM:DETEctor<Trace>:REFerence:MEASStoref<RefMeasurement>
```

class MeastoRefCls

MeastoRef commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RefMeasurement, default value after init: RefMeasurement.Nr1

set(trace=Trace.Default, refMeasurement=RefMeasurement.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:AM:DETEctor<det>:REFerence:MEASStoref<t>
driver.configure.ademod.results.am.detector.reference.meastoRef.set(trace =
↳repcap.Trace.Default, refMeasurement = repcap.RefMeasurement.Default)
```

Sets the reference value to be used for relative demodulation results to the currently measured value on the specified trace for all relative detectors. If necessary, the detectors are activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

param refMeasurement

optional repeated capability selector. Default value: Nr1 (settable in the interface 'MeastoRef')

set_with_opc(trace=Trace.Default, refMeasurement=RefMeasurement.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.am.detector.reference.meastoRef.clone()
```

6.4.1.1.1.5 State

SCPI Commands

```
CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → bool

```
# SCPI: CONFigure:ADEMod:RESults:AM:DETEctor<det>:STATe
value: bool = driver.configure.ademod.results.am.detector.state.get(trace = ↵
↵repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, trace=Trace.Default*) → None

```
# SCPI: CONFigure:ADEMod:RESults:AM:DETEctor<det>:STATe
driver.configure.ademod.results.am.detector.state.set(state = False, trace = ↵
↵repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.4.1.1.2 Fm

class FmCls

Fm commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.fm.clone()
```

Subgroups

6.4.1.1.2.1 Detector<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.configure.ademod.results.fm.detector.repcap_trace_get()
driver.configure.ademod.results.fm.detector.repcap_trace_set(repcap.Trace.Tr1)
```

class DetectorCls

Detector commands group definition. 4 total commands, 3 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.fm.detector.clone()
```

Subgroups

6.4.1.1.2.2 Mode

SCPI Commands

```
CONFigure:ADEMod:RESults:FM:DETEctor<Trace>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → TraceModeE

```
# SCPI: CONFigure:ADEMod:RESults:FM:DETEctor<det>:MODE
value: enums.TraceModeE = driver.configure.ademod.results.fm.detector.mode.
    ↪ get(trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Detector’)

return

mode: WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERAGE The average result is determined over all sweeps.

MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

set(mode: TraceModeE, trace=Trace.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETEctor<det>:MODE
driver.configure.ademod.results.fm.detector.mode.set(mode = enums.TraceModeE.
↳ AVERage, trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param mode

WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERage The average result is determined over all sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.4.1.1.2.3 Reference

SCPI Commands

CONFIGure:ADEMod:RESults:FM:DETEctor<Trace>:REference

class ReferenceCls

Reference commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(trace=Trace.Default) → float

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETEctor<det>:REFERENCE
value: float = driver.configure.ademod.results.fm.detector.reference.get(trace,
↳ repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

ref_value: double value The unit depends on the demodulation type: ACV: V AM: %
FM: Hz PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

set(ref_value: float, trace=Trace.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETEctor<det>:REFERENCE
driver.configure.ademod.results.fm.detector.reference.set(ref_value = 1.0,
↳ trace = repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param ref_value

double value The unit depends on the demodulation type: ACV: V AM: % FM: Hz
PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.fm.detector.reference.clone()
```

Subgroups

6.4.1.1.2.4 MeastoRef<RefMeasurement>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.ademod.results.fm.detector.reference.meastoRef.repcap_
↪refMeasurement_get()
driver.configure.ademod.results.fm.detector.reference.meastoRef.repcap_refMeasurement_
↪set(repcap.RefMeasurement.Nr1)
```

SCPI Commands

```
CONFIGure:ADEMod:RESults:FM:DETECTOR<Trace>:REFerence:MEAStoRef<RefMeasurement>
```

class MeastoRefCls

MeastoRef commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RefMeasurement, default value after init: RefMeasurement.Nr1

set(trace=Trace.Default, refMeasurement=RefMeasurement.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETECTOR<det>:REFerence:MEAStoRef<t>
driver.configure.ademod.results.fm.detector.reference.meastoRef.set(trace = ↪
↪repcap.Trace.Default, refMeasurement = repcap.RefMeasurement.Default)
```

Sets the reference value to be used for relative demodulation results to the currently measured value on the specified trace for all relative detectors. If necessary, the detectors are activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

param refMeasurement

optional repeated capability selector. Default value: Nr1 (settable in the interface 'MeastoRef')

set_with_opc(*trace=Trace.Default, refMeasurement=RefMeasurement.Default, opc_timeout_ms: int = -1*)
→ None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.fm.detector.reference.meastoRef.clone()
```

6.4.1.1.2.5 State**SCPI Commands**

```
CONFIGure:ADEMod:RESults:FM:DETEctor<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → bool

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETEctor<det>:STATe
value: bool = driver.configure.ademod.results.fm.detector.state.get(trace = ↵
↵repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, trace=Trace.Default*) → None

```
# SCPI: CONFIGure:ADEMod:RESults:FM:DETEctor<det>:STATe
driver.configure.ademod.results.fm.detector.state.set(state = False, trace = ↵
↵repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.4.1.1.3 Pm**class PmCls**

Pm commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.pm.clone()
```

Subgroups**6.4.1.1.3.1 Detector<Trace>****RepCap Settings**

```
# Range: Tr1 .. Tr16
rc = driver.configure.ademod.results.pm.detector.repcap_trace_get()
driver.configure.ademod.results.pm.detector.repcap_trace_set(repcap.Trace.Tr1)
```

class DetectorCls

Detector commands group definition. 4 total commands, 3 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.pm.detector.clone()
```

Subgroups**6.4.1.1.3.2 Mode****SCPI Commands**

```
CONFigure:ADEMod:RESults:PM:DETEctor<Trace>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → TraceModeE

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETector<det>:MODE
value: enums.TraceModeE = driver.configure.ademod.results.pm.detector.mode.
↪get(trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

mode: WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERage The average result is determined over all sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

set(mode: TraceModeE, trace=Trace.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETector<det>:MODE
driver.configure.ademod.results.pm.detector.mode.set(mode = enums.TraceModeE.
↪AVERage, trace = repcap.Trace.Default)
```

Defines the mode with which the demodulation result is determined.

param mode

WRITE Overwrite mode: the detector value is overwritten by each sweep. This is the default setting. AVERage The average result is determined over all sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves each result only if the new value is greater than the previous one.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.4.1.1.3.3 Reference

SCPI Commands

```
CONFIGure:ADEMod:RESults:PM:DETector<Trace>:REference
```

class ReferenceCls

Reference commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(trace=Trace.Default) → float

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETector<det>:REference
value: float = driver.configure.ademod.results.pm.detector.reference.get(trace,
↪= repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

ref_value: double value The unit depends on the demodulation type: ACV: V AM: %
FM: Hz PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

set(ref_value: float, trace=Trace.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETEctor<det>:REFerence
driver.configure.ademod.results.pm.detector.reference.set(ref_value = 1.0,
↳ trace = repcap.Trace.Default)
```

Defines the reference value to be used for relative demodulation results and recalculates the results. If necessary, the detector is activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param ref_value

double value The unit depends on the demodulation type: ACV: V AM: % FM: Hz
PM: depends on method RsFswp.Unit.Angle.set setting Unit: RAD

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.pm.detector.reference.clone()
```

Subgroups**6.4.1.1.3.4 MeastoRef<RefMeasurement>****RepCap Settings**

```
# Range: Nr1 .. Nr32
rc = driver.configure.ademod.results.pm.detector.reference.meastoRef.repcap_
↳ refMeasurement_get()
driver.configure.ademod.results.pm.detector.reference.meastoRef.repcap_refMeasurement_
↳ set(repcap.RefMeasurement.Nr1)
```

SCPI Commands

```
CONFIGure:ADEMod:RESults:PM:DETEctor<Trace>:REFerence:MEAStoRef<RefMeasurement>
```

class MeastoRefCls

MeastoRef commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RefMeasurement, default value after init: RefMeasurement.Nr1

set(trace=Trace.Default, refMeasurement=RefMeasurement.Default) → None

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETEctor<det>:REFerence:MEAStoRef<t>
driver.configure.ademod.results.pm.detector.reference.meastoRef.set(trace = ↵
↵repcap.Trace.Default, refMeasurement = repcap.RefMeasurement.Default)
```

Sets the reference value to be used for relative demodulation results to the currently measured value on the specified trace for all relative detectors. If necessary, the detectors are activated. A reference value 0 would provide infinite results and is thus automatically corrected to 0.1.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Detector’)

param refMeasurement

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘MeastoRef’)

set_with_opc(trace=Trace.Default, refMeasurement=RefMeasurement.Default, opc_timeout_ms: int = -1)
→ None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.ademod.results.pm.detector.reference.meastoRef.clone()
```

6.4.1.1.3.5 State

SCPI Commands

```
CONFIGure:ADEMod:RESults:PM:DETEctor<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → bool

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETEctor<det>:STATe
value: bool = driver.configure.ademod.results.pm.detector.state.get(trace = ↵
↵repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Detector’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(*state: bool, trace=Trace.Default*) → None

```
# SCPI: CONFIGure:ADEMod:RESults:PM:DETEctor<det>:STATe
driver.configure.ademod.results.pm.detector.state.set(state = False, trace =
↳repcap.Trace.Default)
```

Activates relative demodulation for the selected detector. If activated, the demodulated result is set in relation to the reference value defined by method RsFswp.Configure.Ademod.Results.Pm.Detector.Reference.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.4.1.1.4 Unit

SCPI Commands

```
CONFIGure:ADEMod:RESults:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → UnitMode

```
# SCPI: CONFIGure:ADEMod:RESults:UNIT
value: enums.UnitMode = driver.configure.ademod.results.unit.get()
```

This command selects the unit for relative demodulation results.

return

unit: PCT | DB

set(*unit: UnitMode*) → None

```
# SCPI: CONFIGure:ADEMod:RESults:UNIT
driver.configure.ademod.results.unit.set(unit = enums.UnitMode.DB)
```

This command selects the unit for relative demodulation results.

param unit

PCT | DB

6.4.2 Generator

class GeneratorCls

Generator commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.generator.clone()
```

Subgroups

6.4.2.1 Connection

class ConnectionCls

Connection commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.generator.connection.clone()
```

Subgroups

6.4.2.1.1 Cstate

SCPI Commands

```
CONFigure:GENerator:CONNection:CState
```

class CstateCls

Cstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: CONFigure:GENerator:CONNection:CState
value: str = driver.configure.generator.connection.cstate.get()
```

No command help available

```
return
    connection_state: No help available
```

6.4.2.1.2 State

SCPI Commands

CONFigure:GENerator:CONNection:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: CONFigure:GENerator:CONNection[:STATe]
value: bool = driver.configure.generator.connection.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: CONFigure:GENerator:CONNection[:STATe]
driver.configure.generator.connection.state.set(state = False)
```

No command help available

param state

No help available

6.4.2.2 IpConnection

class IpConnectionCls

IpConnection commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.generator.ipConnection.clone()
```

Subgroups

6.4.2.2.1 Address

SCPI Commands

CONFigure:GENerator:IPConnection:ADDReSS

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: CONFIGure:GENerator:IPConnection:ADDRESS
value: str = driver.configure.generator.ipConnection.address.get()
```

No command help available

```
return
ip_address: No help available
```

set(ip_address: str) → None

```
# SCPI: CONFIGure:GENerator:IPConnection:ADDRESS
driver.configure.generator.ipConnection.address.set(ip_address = '1')
```

No command help available

```
param ip_address
No help available
```

6.4.2.3 Recording

class RecordingCls

Recording commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.generator.recording.clone()
```

Subgroups

6.4.2.3.1 Combine

SCPI Commands

```
CONFIGure:GENerator:RECORDing:COMBine
```

class CombineCls

Combine commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: CONFIGure:GENerator:RECORDing:COMBine
value: bool = driver.configure.generator.recording.combine.get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: CONFIGure:GENerator:RECORDing:COMbine
driver.configure.generator.recording.combine.set(state = False)
```

No command help available

param state

No help available

6.5 Diagnostic

class DiagnosticCls

Diagnostic commands group definition. 58 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.clone()
```

Subgroups

6.5.1 Hums

SCPI Commands

```
DIAGNOSTIC:HUMS:DELETE:ALL
DIAGNOSTIC:HUMS:SAVE
```

class HumsCls

Hums commands group definition. 25 total commands, 13 Subgroups, 2 group commands

delete_all() → None

```
# SCPI: DIAGNOSTIC:HUMS:DELETE:ALL
driver.diagnostic.hums.delete_all()
```

Deletes the complete HUMS data. This includes device history, device tags, SCPI connections, utilization history and utilizations.

delete_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: DIAGNOSTIC:HUMS:DELETE:ALL
driver.diagnostic.hums.delete_all_with_opc()
```

Deletes the complete HUMS data. This includes device history, device tags, SCPI connections, utilization history and utilizations.

Same as delete_all, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

save(path: str) → None

```
# SCPI: DIAGnostic:HUMS:SAVE
driver.diagnostic.hums.save(path = '1')
```

No command help available

param path

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.clone()
```

Subgroups**6.5.1.1 All****SCPI Commands**

```
DIAGnostic:HUMS:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS[:ALL]
value: bytes = driver.diagnostic.hums.all.get()
```

No command help available

return

information: No help available

6.5.1.2 Bios**SCPI Commands**

```
DIAGnostic:HUMS:BIOS
```

class BiosCls

Bios commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:BIOS
value: bytes = driver.diagnostic.hums.bios.get()
```

No command help available

```
return
    bios_info: No help available
```

6.5.1.3 Device

class DeviceCls

Device commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.device.clone()
```

Subgroups

6.5.1.3.1 History

SCPI Commands

```
DIAGnostic:HUMS:DEvice:HISTory
DIAGnostic:HUMS:DEvice:HISTory:DElete:ALL
```

class HistoryCls

History commands group definition. 2 total commands, 0 Subgroups, 2 group commands

delete_all() → None

```
# SCPI: DIAGnostic:HUMS:DEvice:HISTory:DElete:ALL
driver.diagnostic.hums.device.history.delete_all()
```

No command help available

delete_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: DIAGnostic:HUMS:DEvice:HISTory:DElete:ALL
driver.diagnostic.hums.device.history.delete_all_with_opc()
```

No command help available

Same as delete_all, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

```
param opc_timeout_ms
    Maximum time to wait in milliseconds, valid only for this call.
```

get() → bytes

```
# SCPI: DIAGnostic:HUMS:DEvice:HISTory
value: bytes = driver.diagnostic.hums.device.history.get()
```

No command help available

```

return
    device_history: No help available

```

6.5.1.4 Equipment

SCPI Commands

```
DIAGnostic:HUMS:EQUipment
```

class EquipmentCls

Equipment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```

# SCPI: DIAGnostic:HUMS:EQUipment
value: bytes = driver.diagnostic.hums.equipment.get()

```

No command help available

```

return
    equipment_info: No help available

```

6.5.1.5 FormatPy

SCPI Commands

```
DIAGnostic:HUMS:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → HumsFileFormat

```

# SCPI: DIAGnostic:HUMS:FORMat
value: enums.HumsFileFormat = driver.diagnostic.hums.formatPy.get()

```

Selects the format for the queried HUMS data. You can query the HUMS data either in JSON format or XML format. The defined format affects all other commands that return block data.

```

return
    format_py: No help available

```

set(format_py: HumsFileFormat) → None

```

# SCPI: DIAGnostic:HUMS:FORMat
driver.diagnostic.hums.formatPy.set(format_py = enums.HumsFileFormat.JSON)

```

Selects the format for the queried HUMS data. You can query the HUMS data either in JSON format or XML format. The defined format affects all other commands that return block data.

```

param format_py
    JSON|XML JSON Returns the HUMS data in JSON format. XML Returns the HUMS
    data in XML format.

```

6.5.1.6 Security

SCPI Commands

DIAGnostic:HUMS:SECurity

class SecurityCls

Security commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:SECurity
value: bytes = driver.diagnostic.hums.security.get()
```

No command help available

```
return
    security_info: No help available
```

6.5.1.7 Service

SCPI Commands

DIAGnostic:HUMS:SERVice

class ServiceCls

Service commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:SERVice
value: bytes = driver.diagnostic.hums.service.get()
```

No command help available

```
return
    service_info: No help available
```

6.5.1.8 State

SCPI Commands

DIAGnostic:HUMS:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DIAGnostic:HUMS[:STATe]
value: bool = driver.diagnostic.hums.state.get()
```

Turns the HUMS service and data collection on and off.


```

    return
        state: ON | OFF | 1 | 0

set(state: bool) → None

```

```

# SCPI: DIAGnostic:HUMS[:STATe]
driver.diagnostic.hums.state.set(state = False)

```

Turns the HUMS service and data collection on and off.

```

param state
    ON | OFF | 1 | 0

```

6.5.1.9 Storage

SCPI Commands

```
DIAGnostic:HUMS:STORage
```

class StorageCls

Storage commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```

# SCPI: DIAGnostic:HUMS:STORage
value: bytes = driver.diagnostic.hums.storage.get()

```

No command help available

```

return
    storage_info: No help available

```

6.5.1.10 Sw

SCPI Commands

```
DIAGnostic:HUMS:SW
```

class SwCls

Sw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```

# SCPI: DIAGnostic:HUMS:SW
value: bytes = driver.diagnostic.hums.sw.get()

```

No command help available

```

return
    software_info: No help available

```

6.5.1.11 System

class SystemCls

System commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.system.clone()
```

Subgroups

6.5.1.11.1 Info

SCPI Commands

```
DIAGnostic:HUMS:SYSTem:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:SYSTem:INFO
value: bytes = driver.diagnostic.hums.system.info.get()
```

No command help available

```
return
    system_info: No help available
```

6.5.1.11.2 Status

SCPI Commands

```
DIAGnostic:HUMS:SYSTem:STATus
```

class StatusCls

Status commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:SYSTem:STATus
value: bytes = driver.diagnostic.hums.system.status.get()
```

No command help available

```
return
    system_status: No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.system.status.clone()
```

Subgroups

6.5.1.11.2.1 Summary

SCPI Commands

```
DIAGnostic:HUMS:SYSTem:STATus:SUMMary
```

class SummaryCls

Summary commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SystemStatus

```
# SCPI: DIAGnostic:HUMS:SYSTem:STATus:SUMMary
value: enums.SystemStatus = driver.diagnostic.hums.system.status.summary.get()
```

No command help available

```
return
    system_status: No help available
```

6.5.1.12 Tags

SCPI Commands

```
DIAGnostic:HUMS:TAgS:DELeTe
DIAGnostic:HUMS:TAgS:DELeTe:ALL
```

class TagsCls

Tags commands group definition. 4 total commands, 2 Subgroups, 2 group commands

delete(delete_tag: str) → None

```
# SCPI: DIAGnostic:HUMS:TAgS:DELeTe
driver.diagnostic.hums.tags.delete(delete_tag = '1')
```

Deletes a certain tag you assigned to your instrument, including its key and value.

```
param delete_tag
    ID number of the tag you want to delete. To identify the ID number, query
    all device tags from the system first. For more information, see method RsF-
swp.Diagnostic.Hums.Tags.All.get_
```

delete_all() → None

```
# SCPI: DIAGnostic:HUMS:TAgS:DELeTe:ALL
driver.diagnostic.hums.tags.delete_all()
```

Deletes all key-value tags you have assigned to the instrument.

delete_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: DIAGnostic:HUMS:TAGS:DElete:ALL
driver.diagnostic.hums.tags.delete_all_with_opc()
```

Deletes all key-value tags you have assigned to the instrument.

Same as delete_all, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.tags.clone()
```

Subgroups

6.5.1.12.1 All

SCPI Commands

```
DIAGnostic:HUMS:TAGS:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:TAGS:ALL
value: bytes = driver.diagnostic.hums.tags.all.get()
```

Queries all key-value tags that you have assigned to the instrument. Depending on the set data format, the queried data is either displayed in XML or JSON format. For more information about setting the data format, see method RsFswp.Diagnostic.Hums.FormatPy.set.

return

tags_info: No help available

6.5.1.12.2 Value

SCPI Commands

```
DIAGnostic:HUMS:TAGS:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ValueStruct

Response structure. Fields:

- Idn: float: 0 - 31 ID number of the tag you want to modify or query. To identify the ID number, query all device tags from the system first. For more information, read here [CMDLINK: DIAGnostic:HUMS:TAGS:ALL? CMDLINK].
- Key: str: String containing key name of the queried tag.
- Value: str: String containing value of the queried tag.

get() → ValueStruct

```
# SCPI: DIAGnostic:HUMS:TAGS:VALue
value: ValueStruct = driver.diagnostic.hums.tags.value.get()
```

Adds or modifies a key-value pair (device tag) . The query returns the key-value pair for a given ID or an empty string if the ID is unknown.

return

structure: for return value, see the help for ValueStruct structure arguments.

set(idn: float, key: str, value: str) → None

```
# SCPI: DIAGnostic:HUMS:TAGS:VALue
driver.diagnostic.hums.tags.value.set(idn = 1.0, key = '1', value = '1')
```

Adds or modifies a key-value pair (device tag) . The query returns the key-value pair for a given ID or an empty string if the ID is unknown.

param idn

0 - 31 ID number of the tag you want to modify or query. To identify the ID number, query all device tags from the system first. For more information, read here method **RsFswp.Diagnostic.Hums.Tags.All.get_**.

param key

String containing key name of the queried tag.

param value

String containing value of the queried tag.

6.5.1.13 Utilization**SCPI Commands**

```
DIAGnostic:HUMS:UTILization
```

class UtilizationCls

Utilization commands group definition. 5 total commands, 2 Subgroups, 1 group commands

get() → bytes

```
# SCPI: DIAGnostic:HUMS:UTILization
value: bytes = driver.diagnostic.hums.utilization.get()
```

No command help available

return
utilization: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently  
group2 = driver.diagnostic.hums.utilization.clone()
```

Subgroups

6.5.1.13.1 Activity

SCPI Commands

```
DIAGnostic:HUMS:UTILization:ACTivity
```

class ActivityCls

Activity commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → bool

```
# SCPI: DIAGnostic:HUMS:UTILization:ACTivity  
value: bool = driver.diagnostic.hums.utilization.activity.get()
```

No command help available

return
state: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently  
group2 = driver.diagnostic.hums.utilization.activity.clone()
```

Subgroups

6.5.1.13.1.1 Tracking

class TrackingCls

Tracking commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.hums.utilization.activity.tracking.clone()
```

Subgroups

6.5.1.13.1.2 State

SCPI Commands

```
DIAGnostic:HUMS:UTILization:ACTivity:TRACking:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class StateStruct

Response structure. Fields:

- Idn: float: No parameter help available
- State: bool: No parameter help available

get() → StateStruct

```
# SCPI: DIAGnostic:HUMS:UTILization:ACTivity:TRACking:STATe
value: StateStruct = driver.diagnostic.hums.utilization.activity.tracking.state.
↳ get()
```

No command help available

return

structure: for return value, see the help for StateStruct structure arguments.

set(idn: float, state: bool) → None

```
# SCPI: DIAGnostic:HUMS:UTILization:ACTivity:TRACking:STATe
driver.diagnostic.hums.utilization.activity.tracking.state.set(idn = 1.0, state_
↳ = False)
```

No command help available

param idn

No help available

param state

No help available

6.5.1.13.2 History

SCPI Commands

```
DIAGnostic:HUMS:UTILization:HISTory  
DIAGnostic:HUMS:UTILization:HISTory:DElete:ALL
```

class HistoryCls

History commands group definition. 2 total commands, 0 Subgroups, 2 group commands

delete_all() → None

```
# SCPI: DIAGnostic:HUMS:UTILization:HISTory:DElete:ALL  
driver.diagnostic.hums.utilization.history.delete_all()
```

No command help available

delete_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: DIAGnostic:HUMS:UTILization:HISTory:DElete:ALL  
driver.diagnostic.hums.utilization.history.delete_all_with_opc()
```

No command help available

Same as delete_all, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

get() → bytes

```
# SCPI: DIAGnostic:HUMS:UTILization:HISTory  
value: bytes = driver.diagnostic.hums.utilization.history.get()
```

No command help available

return

util_history: No help available

6.5.2 Info

class InfoCls

Info commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.info.clone()
```

Subgroups

6.5.2.1 Ccount

SCPI Commands

```
DIAGnostic:INFO:CCount
```

class CcountCls

Ccount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(relay: Relay) → int

```
# SCPI: DIAGnostic:INFO:CCount
value: int = driver.diagnostic.info.ccount.get(relay = enums.Relay.AC_enable)
```

This command queries how many switching cycles the individual relays have performed since they were installed.

param relay

ATT5 Mechanical Attenuation 05 DB ATT10 Mechanical Attenuation 10 DB ATT20 Mechanical Attenuation 20 DB ATT40 Mechanical Attenuation 40 DB CAL Mechanical Calibration Source ACDC Mechanical Attenuation Coupling PREamp Preamplifier Bypass PRES Preselector 1: PRESEL RFAB Preselector 1: RFAB PRE Preselector 1: PREAMP30MHZ ATT Preselector 1: ATTINPUT2 INP Preselector 1: INPUT2 **EXT_** Preselector 2: EXT_RELAIS SATT10 | SATT20 | SATT40 Mechanical attenuation (10, 20 and 40 dB) for the optional Signal Source hardware. SCAL DUT bypass (available with the optional Signal Source hardware) .

return

cycles: Number of switching cycles.

6.5.3 Service

class ServiceCls

Service commands group definition. 32 total commands, 12 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.clone()
```

Subgroups

6.5.3.1 BiosInfo

SCPI Commands

```
DIAGnostic:SERvice:BIOSinfo
```

class BiosInfoCls

BiosInfo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SERvice:BIOSinfo
value: str = driver.diagnostic.service.biosInfo.get()
```

This command queries the BIOS version of the CPU board.

return
bios_information: String containing the BIOS version.

6.5.3.2 Calibration

class CalibrationCls

Calibration commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.calibration.clone()
```

Subgroups

6.5.3.2.1 Date

SCPI Commands

```
DIAGnostic:SERvice:CALibration:DATE
```

class DateCls

Date commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:CALibration:DATE
value: str = driver.diagnostic.service.calibration.date.get()
```

Defines last date and time the instrument was calibrated in ISO 8601 format.

return
date: No help available

set(date: str) → None

```
# SCPI: DIAGnostic:SErvice:CALibration:DATE
driver.diagnostic.service.calibration.date.set(date = '1')
```

Defines last date and time the instrument was calibrated in ISO 8601 format.

param date
String containing calibration date of the instrument.

6.5.3.2.2 Due

class DueCls

Due commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.calibration.due.clone()
```

Subgroups

6.5.3.2.2.1 Date

SCPI Commands

```
DIAGnostic:SErvice:CALibration:DUE:DATE
```

class DateCls

Date commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:CALibration:DUE:DATE
value: str = driver.diagnostic.service.calibration.due.date.get()
```

Defines next date and time the instrument needs calibration to be done in ISO 8601 format. The response may be empty in case of no fixed next calibration due.

return
date: No help available

set(date: str) → None

```
# SCPI: DIAGnostic:SERvice:CALibration:DUE:DATE
driver.diagnostic.service.calibration.due.date.set(date = '1')
```

Defines next date and time the instrument needs calibration to be done in ISO 8601 format. The response may be empty in case of no fixed next calibration due.

param date

String containing next calibration due date. An empty string resets the date (= no due date) .

6.5.3.2.2 State

SCPI Commands

```
DIAGnostic:SERvice:CALibration:DUE:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CalibrationState

```
# SCPI: DIAGnostic:SERvice:CALibration:DUE:STATe
value: enums.CalibrationState = driver.diagnostic.service.calibration.due.state.
↳get()
```

No command help available

return

state: No help available

6.5.3.2.3 Interval

SCPI Commands

```
DIAGnostic:SERvice:CALibration:INTERval
```

class IntervalCls

Interval commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SERvice:CALibration:INTERval
value: str = driver.diagnostic.service.calibration.interval.get()
```

This command queries the recommended calibration interval (ISO 8601 duration) .

return

interval: String containing the recommended calibration interval.

set(interval: str) → None

```
# SCPI: DIAGnostic:SErvice:CALibration:INTERval
driver.diagnostic.service.calibration.interval.set(interval = '1')
```

This command queries the recommended calibration interval (ISO 8601 duration) .

param interval

String containing the recommended calibration interval.

6.5.3.3 Date

SCPI Commands

```
DIAGnostic:SErvice:DATE
```

class DateCls

Date commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:DATE
value: str = driver.diagnostic.service.date.get()
```

Defines the last date and time the instrument was serviced (ISO 8601 format) .

return

date: No help available

set(date: str) → None

```
# SCPI: DIAGnostic:SErvice:DATE
driver.diagnostic.service.date.set(date = '1')
```

Defines the last date and time the instrument was serviced (ISO 8601 format) .

param date

String containing last service date.

6.5.3.4 HwInfo

SCPI Commands

```
DIAGnostic:SErvice:HWInfo
```

class HwInfoCls

HwInfo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:HWInfo
value: str = driver.diagnostic.service.hwInfo.get()
```

This command queries hardware information.

return

hardware: String containing the following information for every hardware component.
component: name of the hardware component
serial#: serial number of the component
order#: order number of the component
model: model of the component
code: code of the component
revision: revision of the component
subrevision: subrevision of the component

6.5.3.5 InputPy

class InputPyCls

InputPy commands group definition. 14 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.clone()
```

Subgroups

6.5.3.5.1 AiQ

class AiQCls

AiQ commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.aiq.clone()
```

Subgroups

6.5.3.5.1.1 TypePy

SCPI Commands

```
DIAGnostic:SERvice:INPut:AIQ:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SignalType

```
# SCPI: DIAGnostic:SERvice:INPut:AIQ[:TYPE]
value: enums.SignalType = driver.diagnostic.service.inputPy.aiq.typePy.get()
```

No command help available

return
 signal_type: No help available

set(signal_type: *SignalType*) → None

```
# SCPI: DIAGnostic:SERVice:INPut:AIQ[:TYPE]
driver.diagnostic.service.inputPy.aiq.typePy.set(signal_type = enums.SignalType.
↪AC)
```

No command help available

param signal_type
 No help available

6.5.3.5.2 Emi

class EmiCls

Emi commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.emi.clone()
```

Subgroups

6.5.3.5.2.1 Cfrequency

SCPI Commands

```
DIAGnostic:SERVice:INPut:EMI:CFrequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SERVice:INPut:EMI:CFrequency
value: float = driver.diagnostic.service.inputPy.emi.cfrequency.get()
```

No command help available

return
 frequency: No help available

set(frequency: *float*) → None

```
# SCPI: DIAGnostic:SERVice:INPut:EMI:CFrequency
driver.diagnostic.service.inputPy.emi.cfrequency.set(frequency = 1.0)
```

No command help available

param frequency
No help available

6.5.3.5.2.2 Spectrum

SCPI Commands

DIAGnostic:SERvice:INPut:EMI:SPECTrum

class SpectrumCls

Spectrum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ServiceBandwidth

<pre># SCPI: DIAGnostic:SERvice:INPut:EMI[:SPECTrum] value: enums.ServiceBandwidth = driver.diagnostic.service.inputPy.emi.spectrum. ↳get()</pre>

No command help available

return
spectrum: No help available

set(spectrum: ServiceBandwidth) → None

<pre># SCPI: DIAGnostic:SERvice:INPut:EMI[:SPECTrum] driver.diagnostic.service.inputPy.emi.spectrum.set(spectrum = enums. ↳ServiceBandwidth.BROadband)</pre>
--

No command help available

param spectrum
No help available

6.5.3.5.3 Mc

class McCls

Mc commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

<pre># Create a clone of the original group, that exists independently group2 = driver.diagnostic.service.inputPy.mc.clone()</pre>
--

Subgroups

6.5.3.5.3.1 Cfrequency

SCPI Commands

```
DIAGnostic:SERvice:INPut:MC:CFrequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SERvice:INPut:MC:CFrequency
value: float = driver.diagnostic.service.inputPy.mc.cfrequency.get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: DIAGnostic:SERvice:INPut:MC:CFrequency
driver.diagnostic.service.inputPy.mc.cfrequency.set(frequency = 1.0)
```

No command help available

param frequency

No help available

6.5.3.5.3.2 Distance

SCPI Commands

```
DIAGnostic:SERvice:INPut:MC:DISTance
```

class DistanceCls

Distance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TuningRange

```
# SCPI: DIAGnostic:SERvice:INPut:MC[:DISTance]
value: enums.TuningRange = driver.diagnostic.service.inputPy.mc.distance.get()
```

This command selects the distance of the peaks of the microwave calibration signal for calibration of the YIG filter.

return

bandwidth: WIDE | SMALL SMALL Small offset of combine frequencies. WIDE
Wide offset of combine frequencies.

set(*bandwidth: TuningRange*) → None

```
# SCPI: DIAGnostic:SErvice:INPut:MC[:DISTance]
driver.diagnostic.service.inputPy.mc.distance.set(bandwidth = enums.TuningRange.
↪SMALL)
```

This command selects the distance of the peaks of the microwave calibration signal for calibration of the YIG filter.

param bandwidth

WIDE | SMALL SMALL Small offset of combine frequencies. WIDE Wide offset of combine frequencies.

6.5.3.5.3.3 Range

SCPI Commands

```
DIAGnostic:SErvice:INPut:MC:RANGe
```

class RangeCls

Range commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SErvice:INPut:MC:RANGe
value: float = driver.diagnostic.service.inputPy.mc.range.get()
```

No command help available

return

range_py: No help available

set(*range_py: float*) → None

```
# SCPI: DIAGnostic:SErvice:INPut:MC:RANGe
driver.diagnostic.service.inputPy.mc.range.set(range_py = 1.0)
```

No command help available

param range_py

No help available

6.5.3.5.4 Pulsed

class PulsedCls

Pulsed commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.pulsed.clone()
```

Subgroups

6.5.3.5.4.1 Cfrequency

SCPI Commands

```
DIAGnostic:SERvice:INPut:PULSed:CFrequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SERvice:INPut:PULSed:CFrequency
value: float = driver.diagnostic.service.inputPy.pulsed.cfrequency.get()
```

This command defines the frequency of the calibration signal. Before you can use the command, you have to feed in a calibration signal with method RsFswp.Diagnostic.Service.InputPy.Select.set.

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: DIAGnostic:SERvice:INPut:PULSed:CFrequency
driver.diagnostic.service.inputPy.pulsed.cfrequency.set(frequency = 1.0)
```

This command defines the frequency of the calibration signal. Before you can use the command, you have to feed in a calibration signal with method RsFswp.Diagnostic.Service.InputPy.Select.set.

param frequency

No help available

6.5.3.5.4.2 McFrequency

SCPI Commands

```
DIAGnostic:SERvice:INPut:PULSed:MCFrequency
```

class McFrequencyCls

McFrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SERvice:INPut:PULSed:MCFrequency
value: float = driver.diagnostic.service.inputPy.pulsed.mcFrequency.get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: DIAGnostic:SErvice:INPut:PULSed:MCFrequency
driver.diagnostic.service.inputPy.pulsed.mcFrequency.set(frequency = 1.0)
```

No command help available

param frequency

No help available

6.5.3.5.4.3 WbFrequency

SCPI Commands

```
DIAGnostic:SErvice:INPut:PULSed:WBFrequency
```

class WbFrequencyCls

WbFrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SErvice:INPut:PULSed:WBFrequency
value: float = driver.diagnostic.service.inputPy.pulsed.wbFrequency.get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: DIAGnostic:SErvice:INPut:PULSed:WBFrequency
driver.diagnostic.service.inputPy.pulsed.wbFrequency.set(frequency = 1.0)
```

No command help available

param frequency

No help available

6.5.3.5.5 Rf

class RfCls

Rf commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.rf.clone()
```

Subgroups

6.5.3.5.5.1 Cfrequency

SCPI Commands

```
DIAGnostic:SERvice:INPut:RF:CFRequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SERvice:INPut:RF:CFRequency
value: float = driver.diagnostic.service.inputPy.rf.cfrequency.get()
```

No command help available

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: DIAGnostic:SERvice:INPut:RF:CFRequency
driver.diagnostic.service.inputPy.rf.cfrequency.set(frequency = 1.0)
```

No command help available

param frequency

No help available

6.5.3.5.5.2 Cpath

SCPI Commands

```
DIAGnostic:SERvice:INPut:RF:CPATH
```

class CpathCls

Cpath commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PathToCalibrate

```
# SCPI: DIAGnostic:SERvice:INPut:RF:CPATH
value: enums.PathToCalibrate = driver.diagnostic.service.inputPy.rf.cpath.get()
```

No command help available

return
path_to_calibrate: No help available

set(path_to_calibrate: PathToCalibrate) → None

```
# SCPI: DIAGnostic:SERVice:INPut:RF:CPATH
driver.diagnostic.service.inputPy.rf.cpath.set(path_to_calibrate = enums.
↳ PathToCalibrate.FULL)
```

No command help available

param path_to_calibrate
No help available

6.5.3.5.5.3 Spectrum

SCPI Commands

```
DIAGnostic:SERVice:INPut:RF:SPECTrum
```

class SpectrumCls

Spectrum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ServiceBandwidth

```
# SCPI: DIAGnostic:SERVice:INPut:RF[:SPECTrum]
value: enums.ServiceBandwidth = driver.diagnostic.service.inputPy.rf.spectrum.
↳ get()
```

This command selects the bandwidth of the calibration signal.

return
bandwidth: NARRowband | BROadband NARRowband Narrowband signal for power calibration of the frontend. BROadband Broadband signal for calibration of the IF filter.

set(bandwidth: ServiceBandwidth) → None

```
# SCPI: DIAGnostic:SERVice:INPut:RF[:SPECTrum]
driver.diagnostic.service.inputPy.rf.spectrum.set(bandwidth = enums.
↳ ServiceBandwidth.BROadband)
```

This command selects the bandwidth of the calibration signal.

param bandwidth
NARRowband | BROadband NARRowband Narrowband signal for power calibration of the frontend. BROadband Broadband signal for calibration of the IF filter.

6.5.3.5.6 Select

SCPI Commands

```
DIAGnostic:SErvice:INPut:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*signal: DiagnosticSignal*) → None

```
# SCPI: DIAGnostic:SErvice:INPut[:SElect]
driver.diagnostic.service.inputPy.select.set(signal = enums.DiagnosticSignal.
    ↪ AIQ)
```

This command activates or deactivates the use of an internal calibration signal as input for the R&S FSWP.

param signal

CALibration Uses the calibration signal as RF input. MCALibration Uses the calibration signal for the microwave range as RF input. RF Uses the signal from the RF input. SYNTtwo Uses the calibration signal to check the phase noise of the two synthesizers. A second synthesizer is available as an hardware option.

6.5.3.5.7 SynthTwo

class SynthTwoCls

SynthTwo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.inputPy.synthTwo.clone()
```

Subgroups

6.5.3.5.7.1 Frequency

SCPI Commands

```
DIAGnostic:SErvice:INPut:SYNThtwo:FREquency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DIAGnostic:SErvice:INPut:SYNThtwo[:FREquency]
value: float = driver.diagnostic.service.inputPy.synthTwo.frequency.get()
```

This command selects the frequency which the synthesizers are calibrated for. The command is available when you select the synthesizer as the calibration source with method `Rsfswp.Diagnostic.Service.InputPy.Select.set`.

return
frequency: Unit: Hz

set(frequency: float) → None

```
# SCPI: DIAGnostic:SErvice:INPut:SYNThtwo[:FREQuency]
driver.diagnostic.service.inputPy.synthTwo.frequency.set(frequency = 1.0)
```

This command selects the frequency which the synthesizers are calibrated for. The command is available when you select the synthesizer as the calibration source with method `Rsfswp.Diagnostic.Service.InputPy.Select.set`.

param frequency
Unit: Hz

6.5.3.6 Nsource

SCPI Commands

DIAGnostic:SErvice:NSource

class NsourceCls

Nsource commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DIAGnostic:SErvice:NSource
value: bool = driver.diagnostic.service.nsource.get()
```

This command turns the 28 V supply of the BNC connector labeled [noise source control] on the R&S FSWP on and off.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: DIAGnostic:SErvice:NSource
driver.diagnostic.service.nsource.set(state = False)
```

This command turns the 28 V supply of the BNC connector labeled [noise source control] on the R&S FSWP on and off.

param state
ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.5.3.7 Sfunction

SCPI Commands

```
DIAGnostic:SErvice:SFUNction
```

class SfunctionCls

Sfunction commands group definition. 4 total commands, 2 Subgroups, 1 group commands

get(service_function: str) → str

```
# SCPI: DIAGnostic:SErvice:SFUNction
value: str = driver.diagnostic.service.sfunction.get(service_function = '1')
```

This command starts a service function. The service functions are available after you have entered the level 1 or level 2 system password.

param service_function

String containing the ID of the service function. The ID of the service function is made up out of five numbers, separated by a point. • function group number • board number • function number • parameter 1 (see the Service Manual) • parameter 2 (see the Service Manual)

return

result: String containing the ID of the service function. The ID of the service function is made up out of five numbers, separated by a point. • function group number • board number • function number • parameter 1 (see the Service Manual) • parameter 2 (see the Service Manual)

set(service_function: str) → None

```
# SCPI: DIAGnostic:SErvice:SFUNction
driver.diagnostic.service.sfunction.set(service_function = '1')
```

This command starts a service function. The service functions are available after you have entered the level 1 or level 2 system password.

param service_function

String containing the ID of the service function. The ID of the service function is made up out of five numbers, separated by a point. • function group number • board number • function number • parameter 1 (see the Service Manual) • parameter 2 (see the Service Manual)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.sfunction.clone()
```

Subgroups

6.5.3.7.1 LastResult

SCPI Commands

DIAGnostic:SERvice:SFUNction:LASTresult

class LastResultCls

LastResult commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SERvice:SFUNction:LASTresult
value: str = driver.diagnostic.service.sfunction.lastResult.get()
```

This command queries the results of the most recent service function you have used.

return
result: No help available

6.5.3.7.2 Results

SCPI Commands

DIAGnostic:SERvice:SFUNction:RESults:DElete

class ResultsCls

Results commands group definition. 2 total commands, 1 Subgroups, 1 group commands

delete() → None

```
# SCPI: DIAGnostic:SERvice:SFUNction:RESults:DElete
driver.diagnostic.service.sfunction.results.delete()
```

This command deletes the results in the output buffer for service functions you have used.

delete_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: DIAGnostic:SERvice:SFUNction:RESults:DElete
driver.diagnostic.service.sfunction.results.delete_with_opc()
```

This command deletes the results in the output buffer for service functions you have used.

Same as delete, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.sfunction.results.clone()
```

Subgroups

6.5.3.7.2.1 Save

SCPI Commands

```
DIAGnostic:SERvice:SFUNction:RESults:SAVE
```

class SaveCls

Save commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SERvice:SFUNction:RESults:SAVE
value: str = driver.diagnostic.service.sfunction.results.save.get()
```

This command saves the results in the output buffer for service functions you have used to a file. If no <FileName> parameter is provided, the results are stored to C:/R_S/INSTR/results/ServiceLog.txt. Note that if the buffer is empty, the function returns an error.

return

filename: String containing the path and file name.

set(filename: Optional[str] = None) → None

```
# SCPI: DIAGnostic:SERvice:SFUNction:RESults:SAVE
driver.diagnostic.service.sfunction.results.save.set(filename = '1')
```

This command saves the results in the output buffer for service functions you have used to a file. If no <FileName> parameter is provided, the results are stored to C:/R_S/INSTR/results/ServiceLog.txt. Note that if the buffer is empty, the function returns an error.

param filename

String containing the path and file name.

6.5.3.8 Sinfo

SCPI Commands

```
DIAGnostic:SERvice:SINFO
```

class SinfoCls

Sinfo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:SIInfo
value: str = driver.diagnostic.service.sinfo.get()
```

This command creates a *.zip file with important support information. The *.zip file contains the system configuration information ('device footprint'), the current eeprom data and a screenshot of the screen display (if available). This data is stored to the C:/R_S/INSTR/USER directory on the instrument. As a result of this command, the created file name (including the drive and path) is returned. You can use the resulting file name information as a parameter for the method RsFswp.MassMemory.copy command to store the file on the controller PC. (See method RsFswp.MassMemory.copy) If you contact the Rohde & Schwarz support to get help for a certain problem, send this file to the support in order to identify and solve the problem faster.

return

filename: C:/R_S/INSTR/USER/R&S Device ID_CurrentDate_CurrentTime String containing the drive, path and file name of the created support file, where the file name consists of the following elements: R&S Device ID: The unique R&S device ID indicated in the 'Versions + Options' information CurrentDate: The date on which the file is created (YYYYMMDD) CurrentTime: The time at which the file is created (HHMMSS)

6.5.3.9 SpCheck

class SpCheckCls

SpCheck commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.spCheck.clone()
```

Subgroups

6.5.3.9.1 Execute

SCPI Commands

```
DIAGnostic:SErvice:SPCheck:EXECute
```

class ExecuteCls

Execute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DIAGnostic:SErvice:SPCheck:EXECute
value: bool = driver.diagnostic.service.spCheck.execute.get()
```

No command help available

return

result: No help available

6.5.3.9.2 Result

SCPI Commands

```
DIAGnostic:SERvice:SPCheck:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SERvice:SPCheck:RESult
value: str = driver.diagnostic.service.spCheck.result.get()
```

No command help available

```
return
    result: No help available
```

6.5.3.10 State

SCPI Commands

```
DIAGnostic:SERvice:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ServiceState

```
# SCPI: DIAGnostic:SERvice:STATe
value: enums.ServiceState = driver.diagnostic.service.state.get()
```

No command help available

```
return
    state: No help available
```

6.5.3.11 Stest

class StestCls

Stest commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.service.stest.clone()
```

Subgroups

6.5.3.11.1 Result

SCPI Commands

```
DIAGnostic:SErvice:STESt:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:STESt:RESult
value: str = driver.diagnostic.service.stest.result.get()
```

This command queries the self-test results.

return

results: String of data containing the results. The rows of the self-test result table are separated by commas.

6.5.3.12 VersionInfo

SCPI Commands

```
DIAGnostic:SErvice:VERSInfo
```

class VersionInfoCls

VersionInfo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DIAGnostic:SErvice:VERSInfo
value: str = driver.diagnostic.service.versionInfo.get()
```

This command queries information about the hardware and software components.

return

information: String containing the version of hardware and software components including the types of licenses for installed options.

6.6 Display

class DisplayCls

Display commands group definition. 67 total commands, 17 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.clone()
```

Subgroups

6.6.1 Annotation

class AnnotationCls

Annotation commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.annotation.clone()
```

Subgroups

6.6.1.1 Cbar

SCPI Commands

```
DISPlay:ANNotation:CBAR
```

class CbarCls

Cbar commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:ANNotation:CBAR
value: bool = driver.display.annotation.cbar.get()
```

This command hides or displays the channel bar information.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: DISPlay:ANNotation:CBAR
driver.display.annotation.cbar.set(state = False)
```

This command hides or displays the channel bar information.

param state
ON | OFF | 0 | 1

6.6.1.2 Frequency

SCPI Commands

DISPlay:ANnotation:FREquency

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:ANnotation:FREquency
value: bool = driver.display.annotation.frequency.get()
```

This command turns the label of the x-axis on and off.

return
state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: DISPlay:ANnotation:FREquency
driver.display.annotation.frequency.set(state = False)
```

This command turns the label of the x-axis on and off.

param state
ON | OFF | 0 | 1

6.6.2 Atab

SCPI Commands

DISPlay:ATAB

class AtabCls

Atab commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:ATAB
value: bool = driver.display.atab.get()
```

This command switches between the MultiView tab and the most recently displayed channel. If only one channel is active, this command has no effect.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: DISPlay:ATAB
driver.display.atab.set(state = False)
```

This command switches between the MultiView tab and the most recently displayed channel. If only one channel is active, this command has no effect.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.6.3 Blighting

SCPI Commands

DISPlay:BLIGHting

class BlightingCls

Blighting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: DISPlay:BLIGHting
value: float = driver.display.blighting.get()
```

No command help available

return

brightness: No help available

set(brightness: float) → None

```
# SCPI: DISPlay:BLIGHting
driver.display.blighting.set(brightness = 1.0)
```

No command help available

param brightness

No help available

6.6.4 Cmap<Item>

RepCap Settings

```
# Range: Ix1 .. Ix64
rc = driver.display.cmap.repcap_item_get()
driver.display.cmap.repcap_item_set(repcap.Item.Ix1)
```

class CmapCls

Cmap commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: Item, default value after init: Item.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.cmap.clone()
```

Subgroups

6.6.4.1 Default<Colors>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.display.cmap.default.repcap_colors_get()
driver.display.cmap.default.repcap_colors_set(repcap.Colors.Ix1)
```

SCPI Commands

```
DISPlay:CMAP<Item>:DEFAult<Colors>
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Colors, default value after init: Colors.Ix1

set(*item=Item.Default, colors=Colors.Default*) → None

```
# SCPI: DISPlay:CMAP<it>:DEFAult<ci>
driver.display.cmap.default.set(item = repcap.Item.Default, colors = repcap.
↪Colors.Default)
```

This command selects the color scheme for the display. The query returns the default color scheme.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Cmap’)

param colors

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Default’)

set_with_opc(*item=Item.Default, colors=Colors.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.cmap.default.clone()
```

6.6.4.2 Hsl

SCPI Commands

```
DISPlay:CMAP<Item>:HSL
```

class HslCls

Hsl commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class HslStruct

Response structure. Fields:

- Hue: float: tint Range: 0 to 1
- Sat: float: saturation Range: 0 to 1
- Lum: float: brightness Range: 0 to 1

get(*item=Item.Default*) → HslStruct

```
# SCPI: DISPlay:CMAP<it>:HSL
value: HslStruct = driver.display.cmap.hsl.get(item = repcap.Item.Default)
```

This command selects the color for various screen elements in the display.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

return

structure: for return value, see the help for HslStruct structure arguments.

set(*hue: float, sat: float, lum: float, item=Item.Default*) → None

```
# SCPI: DISPlay:CMAP<it>:HSL
driver.display.cmap.hsl.set(hue = 1.0, sat = 1.0, lum = 1.0, item = repcap.Item.
↪Default)
```

This command selects the color for various screen elements in the display.

param hue

tint Range: 0 to 1

param sat

saturation Range: 0 to 1

param lum

brightness Range: 0 to 1

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

6.6.4.3 Pdefined

SCPI Commands

DISPlay:CMAP<Item>:PDEFined

class PdefinedCls

Pdefined commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*item=Item.Default*) → Color

<pre># SCPI: DISPlay:CMAP<it>:PDEFined value: enums.Color = driver.display.cmap.pdefined.get(item = repcap.Item. ↪Default)</pre>
--

This command selects a predefined color for various screen elements.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Cmap’)

return

color: BLACK | BLUE | BROWN | GREEN | CYAN | RED | MAGenta | YELLOW | WHITE | DGRAY | LGRAY | LBLue | LGREEN | LCYan | LRED | LMAGenta

set(*color: Color, item=Item.Default*) → None

<pre># SCPI: DISPlay:CMAP<it>:PDEFined driver.display.cmap.pdefined.set(color = enums.Color.BLACK, item = repcap.Item. ↪Default)</pre>
--

This command selects a predefined color for various screen elements.

param color

BLACK | BLUE | BROWN | GREEN | CYAN | RED | MAGenta | YELLOW | WHITE | DGRAY | LGRAY | LBLue | LGREEN | LCYan | LRED | LMAGenta

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Cmap’)

6.6.5 Faccess

class FaccessCls

Faccess commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.faccess.clone()
```

Subgroups

6.6.5.1 Position

SCPI Commands

```
DISPlay:FACcEss:POSiTion
```

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DisplayPosition

```
# SCPI: DISPlay:FACcEss:POSiTion
value: enums.DisplayPosition = driver.display.faccess.position.get()
```

No command help available

return
position: No help available

set(position: DisplayPosition) → None

```
# SCPI: DISPlay:FACcEss:POSiTion
driver.display.faccess.position.set(position = enums.DisplayPosition.BOTTom)
```

No command help available

param position
No help available

6.6.6 FormatPy

SCPI Commands

```
DISPlay:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DisplayFormat

```
# SCPI: DISPlay:FORMat
value: enums.DisplayFormat = driver.display.formatPy.get()
```

This command determines which tab is displayed.

return

format_py: SPlit Displays the MultiView tab with an overview of all active channels (See 'R&S MultiView') . SINGle Displays the measurement channel that was previously focused.

set(format_py: DisplayFormat) → None

```
# SCPI: DISPlay:FORMat
driver.display.formatPy.set(format_py = enums.DisplayFormat.SINGLE)
```

This command determines which tab is displayed.

param format_py

SPlit Displays the MultiView tab with an overview of all active channels (See 'R&S MultiView') . SINGle Displays the measurement channel that was previously focused.

6.6.7 Item

class ItemCls

Item commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.item.clone()
```

Subgroups

6.6.7.1 State

SCPI Commands

```
DISPlay:ITERm:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:ITERm[:STATe]
value: bool = driver.display.item.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: DISPlay:ITERm[:STATe]
driver.display.item.state.set(state = False)
```

No command help available

param state

No help available

6.6.8 Logo

SCPI Commands

DISPlay:LOGO

class LogoCls

Logo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:LOGO
value: bool = driver.display.logo.get()
```

Activates/deactivates the printout of the Rohde & Schwarz company logo at the top of each page.

return

state: 1 | 0 | ON | OFF 1 | ON Logo is printed. 0 | OFF Logo is not printed.

set(state: bool) → None

```
# SCPI: DISPlay:LOGO
driver.display.logo.set(state = False)
```

Activates/deactivates the printout of the Rohde & Schwarz company logo at the top of each page.

param state

1 | 0 | ON | OFF 1 | ON Logo is printed. 0 | OFF Logo is not printed.

6.6.9 Minfo

class MinfoCls

Minfo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.minfo.clone()
```

Subgroups

6.6.9.1 State

SCPI Commands

DISPlay:MINFo:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:MINFo[:STATe]
value: bool = driver.display.minfo.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: DISPlay:MINFo[:STATe]
driver.display.minfo.state.set(state = False)
```

No command help available

param state

No help available

6.6.10 PreSelector

class PreSelectorCls

PreSelector commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.preSelector.clone()
```

Subgroups

6.6.10.1 Position

SCPI Commands

DISPlay:PRESelector:POSition

class PositionCls

Position commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DisplayPosition

```
# SCPI: DISPlay:PRESelector:POsition
value: enums.DisplayPosition = driver.display.preSelector.position.get()
```

No command help available

```
return
    position: No help available
```

set(position: DisplayPosition) → None

```
# SCPI: DISPlay:PRESelector:POsition
driver.display.preSelector.position.set(position = enums.DisplayPosition.BOTTom)
```

No command help available

```
param position
    No help available
```

6.6.11 Sbar

class SbarCls

Sbar commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.sbar.clone()
```

Subgroups

6.6.11.1 State

SCPI Commands

```
DISPlay:SBAR:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:SBAR[:STATe]
value: bool = driver.display.sbar.state.get()
```

This command turns the status bar on and off.

```
return
    state: ON | OFF | 0 | 1
```

set(*state: bool*) → None

```
# SCPI: DISPlay:SBAR[:STATe]
driver.display.sbar.state.set(state = False)
```

This command turns the status bar on and off.

param state
ON | OFF | 0 | 1

6.6.12 Skeys

class SkeysCls

Skeys commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.skeys.clone()
```

Subgroups

6.6.12.1 State

SCPI Commands

```
DISPlay:SKEYs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:SKEYs[:STATe]
value: bool = driver.display.skeys.state.get()
```

This command turns the softkey bar on and off.

return
state: ON | OFF | 0 | 1

set(*state: bool*) → None

```
# SCPI: DISPlay:SKEYs[:STATe]
driver.display.skeys.state.set(state = False)
```

This command turns the softkey bar on and off.

param state
ON | OFF | 0 | 1

6.6.13 Tbar

class TbarCls

Tbar commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.tbar.clone()
```

Subgroups

6.6.13.1 State

SCPI Commands

```
DISPlay:TBAR:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: DISPlay:TBAR[:STATe]
value: bool = driver.display.tbar.state.get()
```

This command turns the toolbar on or off.

return
state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: DISPlay:TBAR[:STATe]
driver.display.tbar.state.set(state = False)
```

This command turns the toolbar on or off.

param state
ON | OFF | 1 | 0

6.6.14 Theme

class ThemeCls

Theme commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.theme.clone()
```

Subgroups

6.6.14.1 Catalog

SCPI Commands

```
DISPlay:THEMe:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: DISPlay:THEMe:CATalog
value: str = driver.display.theme.catalog.get()
```

This command queries all available display themes.

return

themes: String containing all available display themes.

6.6.14.2 Select

SCPI Commands

```
DISPlay:THEMe:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(theme: str) → None

```
# SCPI: DISPlay:THEMe:SElect
driver.display.theme.select.set(theme = '1')
```

This command selects the display theme.

param theme

String containing the name of the theme.

6.6.15 Touchscreen

class TouchscreenCls

Touchscreen commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.touchscreen.clone()
```

Subgroups

6.6.15.1 State

SCPI Commands

```
DISPlay:TOUCHscreen:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TouchscreenState

```
# SCPI: DISPlay:TOUCHscreen[:STATe]
value: enums.TouchscreenState = driver.display.touchscreen.state.get()
```

This command controls the touch screen functionality.

return

state: ON | FRAME | OFF ON | 1 Touch screen is active for entire screen OFF | 0 Touch screen is inactivate for entire screen FRAME Touch screen is inactivate for the diagram area of the screen, but active for softkeys, toolbars and menus.

set(state: TouchscreenState) → None

```
# SCPI: DISPlay:TOUCHscreen[:STATe]
driver.display.touchscreen.state.set(state = enums.TouchscreenState.FRAME)
```

This command controls the touch screen functionality.

param state

ON | FRAME | OFF ON | 1 Touch screen is active for entire screen OFF | 0 Touch screen is inactivate for entire screen FRAME Touch screen is inactivate for the diagram area of the screen, but active for softkeys, toolbars and menus.

6.6.16 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.display.window.repcap_window_get()
driver.display.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 47 total commands, 9 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.clone()
```

Subgroups

6.6.16.1 Minfo

class MinfoCls

Minfo commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.minfo.clone()
```

Subgroups

6.6.16.1.1 State

SCPI Commands

```
DISPlay:WINDow<Window>:MINFo:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
value: bool = driver.display.window.minfo.state.get(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:MINFo[:STATe]
driver.display.window.minfo.state.set(state = False, window = repcap.Window.
↳Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.2 Mtable

SCPI Commands

```
DISPlay:WINDow<Window>:MTABLE
```

class MtableCls

Mtable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → AutoMode

```
# SCPI: DISPlay[:WINDow<n>]:MTABLE
value: enums.AutoMode = driver.display.window.mtable.get(window = repcap.Window.
↳Default)
```

This command turns the marker table on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

display_mode: ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table.
AUTO Turns on the marker table if 3 or more markers are active.

set(display_mode: AutoMode, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:MTABLE
driver.display.window.mtable.set(display_mode = enums.AutoMode.AUTO, window =
↳repcap.Window.Default)
```

This command turns the marker table on and off.

param display_mode

ON | 1 Turns on the marker table. OFF | 0 Turns off the marker table. AUTO Turns on the marker table if 3 or more markers are active.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.3 Size

SCPI Commands

`DISPlay:WINDow<Window>:SIZE`

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → Size

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
value: enums.Size = driver.display.window.size.get(window = repcap.Window.
↳Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` command (see method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set`).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

window_size: No help available

set(window_size: Size, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SIZE
driver.display.window.size.set(window_size = enums.Size.LARGE, window = repcap.
↳Window.Default)
```

This command maximizes the size of the selected result display window temporarily. To change the size of several windows on the screen permanently, use the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` command (see method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set`).

param window_size

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.4 Spectrogram

class SpectrogramCls

Spectrogram commands group definition. 5 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.spectrogram.clone()
```

Subgroups

6.6.16.4.1 Color

class ColorCls

Color commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.spectrogram.color.clone()
```

Subgroups

6.6.16.4.1.1 Default

SCPI Commands

```
DISPlay:WINDow<Window>:SPECTrogram:COLor:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor:DEFault
driver.display.window.spectrogram.color.default.set(window = repcap.Window.
↳Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

set_with_opc(window=Window.Default, opc_timeout_ms: int = -1) → None

6.6.16.4.1.2 Lower

SCPI Commands

DISPlay:WINDow<Window>:SPECTrogram:COLor:LOWer

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor:LOWer
value: float = driver.display.window.spectrogram.color.lower.get(window = ↵
↵repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

percentage: No help available

set(percentage: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor:LOWer
driver.display.window.spectrogram.color.lower.set(percentage = 1.0, window = ↵
↵repcap.Window.Default)
```

No command help available

param percentage

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.4.1.3 Shape

SCPI Commands

DISPlay:WINDow<Window>:SPECTrogram:COLor:SHAPE

class ShapeCls

Shape commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:SPECTrogram:COLor:SHAPE
value: float = driver.display.window.spectrogram.color.shape.get(window = ↵
↵repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

shape: No help available

set(shape: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPEctrogram:COLor:SHApe
driver.display.window.spectrogram.color.shape.set(shape = 1.0, window = repcap.
↳Window.Default)
```

No command help available

param shape

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.4.1.4 Style

SCPI Commands

```
DISPlay:WINDow<Window>:SPEctrogram:COLor:STYLE
```

class StyleCls

Style commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → ColorSchemeA

```
# SCPI: DISPlay[:WINDow<n>]:SPEctrogram:COLor[:STYLE]
value: enums.ColorSchemeA = driver.display.window.spectrogram.color.style.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

color_scheme: No help available

set(color_scheme: ColorSchemeA, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPEctrogram:COLor[:STYLE]
driver.display.window.spectrogram.color.style.set(color_scheme = enums.
↳ColorSchemeA.COLD, window = repcap.Window.Default)
```

No command help available

param color_scheme

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.4.1.5 Upper**SCPI Commands**

DISPlay:WINDow<Window>:SPEctrogram:COLor:UPPer

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:SPEctrogram:COLor:UPPer
value: float = driver.display.window.spectrogram.color.upper.get(window = ↵
↵repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

percentage: No help available

set(percentage: float, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:SPEctrogram:COLor:UPPer
driver.display.window.spectrogram.color.upper.set(percentage = 1.0, window = ↵
↵repcap.Window.Default)
```

No command help available

param percentage

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.5 State**SCPI Commands**

DISPlay:WINDow<Window>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPLAY[:WINDOW<n>]:STATE
value: bool = driver.display.window.state.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPLAY[:WINDOW<n>]:STATE
driver.display.window.state.set(state = False, window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.6 Statistics

class StatisticsCls

Statistics commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.statistics.clone()
```

Subgroups

6.6.16.6.1 CumulativeDistribFnc

class CumulativeDistribFncCls

CumulativeDistribFnc commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.statistics.cumulativeDistribFnc.clone()
```

Subgroups

6.6.16.6.1.1 Gauss

SCPI Commands

```
DISPlay:WINDow<Window>:STATistics:CCDF:GAUSs
```

class GaussCls

Gauss commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:STATistics:CCDF:GAUSs
value: bool = driver.display.window.statistics.cumulativeDistribFnc.gauss.
↳get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: No help available

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:STATistics:CCDF:GAUSs
driver.display.window.statistics.cumulativeDistribFnc.gauss.set(state = False,
↳window = repcap.Window.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.7 Subwindow<SubWindow>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.display.window.subwindow.repcap_subWindow_get()
driver.display.window.subwindow.repcap_subWindow_set(repcap.SubWindow.Nr1)
```

class SubwindowCls

Subwindow commands group definition. 18 total commands, 2 Subgroups, 0 group commands Repeated Capability: SubWindow, default value after init: SubWindow.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.clone()
```

Subgroups

6.6.16.7.1 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.display.window.subwindow.trace.repcap_trace_get()
driver.display.window.subwindow.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 14 total commands, 5 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.clone()
```

Subgroups

6.6.16.7.1.1 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:MODE
```

class ModeCls

Mode commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → TraceModeF

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:MODE
value: enums.TraceModeF = driver.display.window.subwindow.trace.mode.get(window↵
↵= repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.
↵Trace.Default)
```

This command selects the trace mode. If necessary, the selected trace is also activated. For max hold, min hold or average trace mode, you can set the number of single measurements with [SENSe:]SWEep:COUNT. Note that synchronization to the end of the measurement is possible only in single sweep mode. For more information, see ‘Analyzing several traces - trace mode’.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Sub-window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

trace_mode: WRITE (default:) Overwrite mode: the trace is overwritten by each sweep. AVERage The average is formed over several sweeps. The ‘Sweep/Average Count’ determines the number of averaging procedures. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is greater than the previous one. MINHold The minimum value is determined from several measurements and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is lower than the previous one. VIEW The current contents of the trace memory are frozen and displayed. BLANK Hides the selected trace. WRHold The trace is overwritten when new data is available, but only after all cross-correlation operations defined for a half decade are done.

set(trace_mode: TraceModeF, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:MODE
driver.display.window.subwindow.trace.mode.set(trace_mode = enums.TraceModeF.
↵AVERage, window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↵trace = repcap.Trace.Default)
```

This command selects the trace mode. If necessary, the selected trace is also activated. For max hold, min hold or average trace mode, you can set the number of single measurements with [SENSe:]SWEep:COUNT. Note that synchronization to the end of the measurement is possible only in single sweep mode. For more information, see ‘Analyzing several traces - trace mode’.

param trace_mode

WRITE (default:) Overwrite mode: the trace is overwritten by each sweep. AVERage The average is formed over several sweeps. The ‘Sweep/Average Count’ determines the number of averaging procedures. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is greater than the previous one. MINHold The minimum value is determined from several measurements and displayed. The R&S

FSWP saves the sweep result in the trace memory only if the new value is lower than the previous one. **VIEW** The current contents of the trace memory are frozen and displayed. **BLANK** Hides the selected trace. **WRHold** The trace is overwritten when new data is available, but only after all cross-correlation operations defined for a half decade are done.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.mode.clone()
```

Subgroups

6.6.16.7.1.2 Hcontinuous

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:MODE:HCONtinuous
```

class HcontinuousCls

Hcontinuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:MODE:HCONtinuous
value: bool = driver.display.window.subwindow.trace.mode.hcontinuous.get(window,
↳= repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.
↳Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(state: bool, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:MODE:HCONtinuous
driver.display.window.subwindow.trace.mode.hcontinuous.set(state = False,
↪ window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace =
↪ repcap.Trace.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.3 Select

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:SElect
driver.display.window.subwindow.trace.select.set(window = repcap.Window.Default,
↪ subWindow = repcap.SubWindow.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

```
set_with_opc(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default,
             opc_timeout_ms: int = -1) → None
```

6.6.16.7.1.4 State

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

```
get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → bool
```

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>[:STATe]
value: bool = driver.display.window.subwindow.trace.state.get(window = repcap.
↳Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.Trace.
↳Default)
```

This command turns a trace on and off. The measurement continues in the background.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

```
set(state: bool, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None
```

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>[:STATe]
driver.display.window.subwindow.trace.state.set(state = False, window = repcap.
↳Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.Trace.
↳Default)
```

This command turns a trace on and off. The measurement continues in the background.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.5 X**class XCls**

X commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.x.clone()
```

Subgroups**6.6.16.7.1.6 Spacing****SCPI Commands**

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:X:SPACing
```

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → ScalingMode

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:X:SPACing
value: enums.ScalingMode = driver.display.window.subwindow.trace.x.spacing.
↪get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↪trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

scale: No help available

set(scale: ScalingMode, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default)
→ None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:X:SPACing
driver.display.window.subwindow.trace.x.spacing.set(scale = enums.ScalingMode.
↳LINear, window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,↳
↳trace = repcap.Trace.Default)
```

No command help available

param scale

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.7 Y

class YCls

Y commands group definition. 9 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.y.clone()
```

Subgroups

6.6.16.7.1.8 Scale

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWindow<SubWindow>:TRACe<Trace>:Y:SCALE
```

class ScaleCls

Scale commands group definition. 8 total commands, 6 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:Y[:SCALE]
value: float = driver.display.window.subwindow.trace.y.scale.get(window =↳
↳repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.
↳Trace.Default)
```

This command defines the display range of the y-axis (for all traces) . Note that the command works only for a logarithmic scaling. You can select the scaling with method `RsFswp.Display.Window.Subwindow.Trace.Y.Spacing.set`.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

range_py: Range: 1 dB to 200 dB, Unit: HZ

set(range_py: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALe]
driver.display.window.subwindow.trace.y.scale.set(range_py = 1.0, window =
↳repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.
↳Trace.Default)
```

This command defines the display range of the y-axis (for all traces) . Note that the command works only for a logarithmic scaling. You can select the scaling with method `RsFswp.Display.Window.Subwindow.Trace.Y.Spacing.set`.

param range_py

Range: 1 dB to 200 dB, Unit: HZ

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.y.scale.clone()
```

Subgroups

6.6.16.7.1.9 Auto

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALE:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*event: EventOnce, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:AUTO
driver.display.window.subwindow.trace.y.scale.auto.set(event = enums.EventOnce.
↳ ONCE, window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↳ trace = repcap.Trace.Default)
```

Automatic scaling of the y-axis is performed once, then switched off again (for all traces) .

param event

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.10 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALE:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default*) → ReferenceMode

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:MODE
value: enums.ReferenceMode = driver.display.window.subwindow.trace.y.scale.mode.
↳ get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↳ trace = repcap.Trace.Default)
```

This command selects the type of scaling of the y-axis (for all traces) . When the display update during remote control is off, this command has no immediate effect.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: ABSolute absolute scaling of the y-axis RELative relative scaling of the y-axis

set(mode: *ReferenceMode*, window=*Window.Default*, subWindow=*SubWindow.Default*, trace=*Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALe]:MODE
driver.display.window.subwindow.trace.y.scale.mode.set(mode = enums.
↳ReferenceMode.ABSolute, window = repcap.Window.Default, subWindow = repcap.
↳SubWindow.Default, trace = repcap.Trace.Default)
```

This command selects the type of scaling of the y-axis (for all traces) . When the display update during remote control is off, this command has no immediate effect.

param mode

ABSolute absolute scaling of the y-axis RELative relative scaling of the y-axis

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.11 Pdivision

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALe:PDIVision
```

class PdivisionCls

Pdivision commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*, subWindow=*SubWindow.Default*, trace=*Trace.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALe]:PDIVision
value: float = driver.display.window.subwindow.trace.y.scale.pdivision.
↳get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↳trace = repcap.Trace.Default)
```


This remote command determines the grid spacing on the Y-axis for all diagrams, where possible. In spectrum displays, for example, this command is not available.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

grid_spacing: No help available

set(grid_spacing: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default)
→ None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:Y[:SCALE]:PDIVision
driver.display.window.subwindow.trace.y.scale.pdivision.set(grid_spacing = 1.0,
↳window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace =
↳repcap.Trace.Default)
```

This remote command determines the grid spacing on the Y-axis for all diagrams, where possible. In spectrum displays, for example, this command is not available.

param grid_spacing

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.12 RefLevel

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWindow<SubWindow>:TRACe<Trace>:Y:SCALE:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:Y[:SCALe]:RLEVel
value: float = driver.display.window.subwindow.trace.y.scale.refLevel.
↳ get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,↳
↳ trace = repcap.Trace.Default)
```

This command defines the reference level (for all traces in all windows) . With a reference level offset 0, the value range of the reference level is modified by the offset.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

scale: No help available

set(scale: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:Y[:SCALe]:RLEVel
driver.display.window.subwindow.trace.y.scale.refLevel.set(scale = 1.0, window↳
↳ = repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap.↳
↳ Trace.Default)
```

This command defines the reference level (for all traces in all windows) . With a reference level offset 0, the value range of the reference level is modified by the offset.

param scale

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.trace.y.scale.refLevel.clone()
```

Subgroups

6.6.16.7.1.13 Offset

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALE:RLEVel:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:RLEVel:OFFSet
value: float = driver.display.window.subwindow.trace.y.scale.refLevel.offset.
↪get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, ↪
↪trace = repcap.Trace.Default)
```

This command defines a reference level offset (for all traces in all windows).

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

offset: Range: -200 dB to 200 dB, Unit: DB

set(offset: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:RLEVel:OFFSet
driver.display.window.subwindow.trace.y.scale.refLevel.offset.set(offset = 1.0, ↪
↪window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = ↪
↪repcap.Trace.Default)
```

This command defines a reference level offset (for all traces in all windows).

param offset

Range: -200 dB to 200 dB, Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Sub-window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.7.1.14 RefPosition**SCPI Commands**

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALe:RPOSition
```

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALe]:RPOSition
value: float = driver.display.window.subwindow.trace.y.scale.refPosition.
↳ get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↳ trace = repcap.Trace.Default)
```

This command defines the vertical position of the reference level on the display grid (for all traces) . The R&S FSWP adjusts the scaling of the y-axis accordingly. For measurements with the optional external generator control, the command defines the position of the reference value.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Sub-window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

position: No help available

set(position: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALe]:RPOSition
driver.display.window.subwindow.trace.y.scale.refPosition.set(position = 1.0,
↳ window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace =
↳ repcap.Trace.Default)
```

This command defines the vertical position of the reference level on the display grid (for all traces) . The R&S FSWP adjusts the scaling of the y-axis accordingly. For measurements with the optional external generator control, the command defines the position of the reference value.

param position

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.15 Rvalue**SCPI Commands**

DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SCALE:RVALue

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default*) → float

<pre># SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:RVALue value: float = driver.display.window.subwindow.trace.y.scale.rvalue.get(window_ ↪= repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap. ↪Trace.Default)</pre>

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

value: Unit: DB

set(*value: float, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default*) → None

<pre># SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y[:SCALE]:RVALue driver.display.window.subwindow.trace.y.scale.rvalue.set(value = 1.0, window = ↪repcap.Window.Default, subWindow = repcap.SubWindow.Default, trace = repcap. ↪Trace.Default)</pre>
--

This command defines the reference value assigned to the reference position in the specified window. Separate reference values are maintained for the various displays.

param value

Unit: DB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.1.16 Spacing

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:TRACe<Trace>:Y:SPACing
```

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → SpacingY

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:TRACe<t>:Y:SPACing
value: enums.SpacingY = driver.display.window.subwindow.trace.y.spacing.
↪get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default, ↪
↪trace = repcap.Trace.Default)
```

This command selects the scaling of the y-axis (for all traces, <t> is irrelevant) . For AF spectrum displays, only the parameters 'LINear' and 'LOGarithmic' are permitted.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

scaling_type: LOGarithmic Logarithmic scaling. LINear Linear scaling in %. LDB Linear scaling in the specified unit. PERCent Linear scaling in %.

set(scaling_type: SpacingY, window=Window.Default, subWindow=SubWindow.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWindow<w>]:TRACe<t>:Y:SPACing
driver.display.window.subwindow.trace.y.spacing.set(scaling_type = enums.
↳SpacingY.LDB, window = repcap.Window.Default, subWindow = repcap.SubWindow.
↳Default, trace = repcap.Trace.Default)
```

This command selects the scaling of the y-axis (for all traces, <t> is irrelevant) . For AF spectrum displays, only the parameters 'LINear' and 'LOGarithmic' are permitted.

param scaling_type

LOGarithmic Logarithmic scaling. LINear Linear scaling in %. LDB Linear scaling in the specified unit. PERCent Linear scaling in %.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.7.2 Zoom

class ZoomCls

Zoom commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.zoom.clone()
```

Subgroups

6.6.16.7.2.1 Area

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWindow<SubWindow>:ZOOM:AREA
```

class AreaCls

Area commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class AreaStruct

Response structure. Fields:

- X_1: float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

- **Y_1:** float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT
- **X_2:** float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT
- **Y_2:** float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

get(window=Window.Default, subWindow=SubWindow.Default) → AreaStruct

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:AREA
value: AreaStruct = driver.display.window.subwindow.zoom.area.get(window = ↵
↵repcap.Window.Default, subWindow = repcap.SubWindow.Default)
```

This command defines the zoom area. To define a zoom area, you first have to turn the zoom on.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

return

structure: for return value, see the help for AreaStruct structure arguments.

set(x_1: float, y_1: float, x_2: float, y_2: float, window=Window.Default, subWindow=SubWindow.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:AREA
driver.display.window.subwindow.zoom.area.set(x_1 = 1.0, y_1 = 1.0, x_2 = 1.0, ↵
↵y_2 = 1.0, window = repcap.Window.Default, subWindow = repcap.SubWindow.
↵Default)
```

This command defines the zoom area. To define a zoom area, you first have to turn the zoom on.

param x_1

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param y_1

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param x_2

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param y_2

Diagram coordinates in % of the complete diagram that define the zoom area. The

lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

6.6.16.7.2.2 Multiple<ZoomWindow>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.display.window.subwindow.zoom.multiple.repcap_zoomWindow_get()
driver.display.window.subwindow.zoom.multiple.repcap_zoomWindow_set(repcap.ZoomWindow.
↪Nr1)
```

class MultipleCls

Multiple commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: ZoomWindow, default value after init: ZoomWindow.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.subwindow.zoom.multiple.clone()
```

Subgroups

6.6.16.7.2.3 Area

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:ZOOM:MUlTiple<ZoomWindow>:AREA
```

class AreaCls

Area commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class AreaStruct

Response structure. Fields:

- X_1: float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT
- Y_1: float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

- **X_2:** float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT
- **Y_2:** float: Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

get(window=Window.Default, subWindow=SubWindow.Default, zoomWindow=ZoomWindow.Default) → AreaStruct

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:MULTiple<zn>:AREA
value: AreaStruct = driver.display.window.subwindow.zoom.multiple.area.
↳ get(window = repcap.Window.Default, subWindow = repcap.SubWindow.Default,
↳ zoomWindow = repcap.ZoomWindow.Default)
```

This command defines the zoom area for a multiple zoom. To define a zoom area, you first have to turn the zoom on.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param zoomWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Multiple')

return

structure: for return value, see the help for AreaStruct structure arguments.

set(x_1: float, y_1: float, x_2: float, y_2: float, window=Window.Default, subWindow=SubWindow.Default, zoomWindow=ZoomWindow.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:MULTiple<zn>:AREA
driver.display.window.subwindow.zoom.multiple.area.set(x_1 = 1.0, y_1 = 1.0, x_
↳ 2 = 1.0, y_2 = 1.0, window = repcap.Window.Default, subWindow = repcap.
↳ SubWindow.Default, zoomWindow = repcap.ZoomWindow.Default)
```

This command defines the zoom area for a multiple zoom. To define a zoom area, you first have to turn the zoom on.

param x_1

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param y_1

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param x_2

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param y_2

Diagram coordinates in % of the complete diagram that define the zoom area. The lower left corner is the origin of coordinate system. The upper right corner is the end point of the system. Range: 0 to 100, Unit: PCT

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param zoomWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Multiple')

6.6.16.7.2.4 State**SCPI Commands**

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:ZOOM:MULTiple<ZoomWindow>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default, zoomWindow=ZoomWindow.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:MULTiple<zn>[:STATe]
value: bool = driver.display.window.subwindow.zoom.multiple.state.get(window =
↳repcap.Window.Default, subWindow = repcap.SubWindow.Default, zoomWindow =
↳repcap.ZoomWindow.Default)
```

This command turns the multiple zoom on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param zoomWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Multiple')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, subWindow=SubWindow.Default, zoomWindow=ZoomWindow.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM:MULTiple<zn>[:STATe]
driver.display.window.subwindow.zoom.multiple.state.set(state = False, window =
↳repcap.Window.Default, subWindow = repcap.SubWindow.Default, zoomWindow =
↳repcap.ZoomWindow.Default)
```

This command turns the multiple zoom on and off.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

param zoomWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Multiple')

6.6.16.7.2.5 State

SCPI Commands

```
DISPlay:WINDow<Window>:SUBWIndow<SubWindow>:ZOOM:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, subWindow=SubWindow.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM[:STATe]
value: bool = driver.display.window.subwindow.zoom.state.get(window = repcap.
↳Window.Default, subWindow = repcap.SubWindow.Default)
```

This command turns the zoom on and off.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, window=Window.Default, subWindow=SubWindow.Default) → None

```
# SCPI: DISPlay[:WINDow<n>][:SUBWIndow<w>]:ZOOM[:STATe]
driver.display.window.subwindow.zoom.state.set(state = False, window = repcap.
↳Window.Default, subWindow = repcap.SubWindow.Default)
```

This command turns the zoom on and off.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param subWindow

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sub-window')

6.6.16.8 Time

SCPI Commands

```
DISPlay:WINDow<Window>:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TIME
value: bool = driver.display.window.time.get(window = repcap.Window.Default)
```

This command adds or removes the date and time from the display.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TIME
driver.display.window.time.set(state = False, window = repcap.Window.Default)
```

This command adds or removes the date and time from the display.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.time.clone()
```

Subgroups

6.6.16.8.1 FormatPy

SCPI Commands

```
DISPlay:WINDow<Window>:TIME:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → TimeFormat

```
# SCPI: DISPlay[:WINDow<n>]:TIME:FORMat
value: enums.TimeFormat = driver.display.window.time.formatPy.get(window = ↵
↵repcap.Window.Default)
```

This command selects the time and date format.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

format_py: US | DE | ISO DE dd.mm.yyyy hh:mm:ss 24 hour format. US mm/dd/yyyy hh:mm:ss 12 hour format. ISO yyyy-mm-dd hh:mm:ss 24 hour format.

set(format_py: TimeFormat, window=Window.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TIME:FORMat
driver.display.window.time.formatPy.set(format_py = enums.TimeFormat.DE, window ↵
↵= repcap.Window.Default)
```

This command selects the time and date format.

param format_py

US | DE | ISO DE dd.mm.yyyy hh:mm:ss 24 hour format. US mm/dd/yyyy hh:mm:ss 12 hour format. ISO yyyy-mm-dd hh:mm:ss 24 hour format.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.6.16.9 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.display.window.trace.repcap_trace_get()
driver.display.window.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 17 total commands, 5 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.clone()
```

Subgroups

6.6.16.9.1 Length

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:LENGth
value: float = driver.display.window.trace.length.get(window = repcap.Window.
↳Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

trace_length: No help available

6.6.16.9.2 Mode

SCPI Commands

`DISPlay:WINDow<Window>:TRACe<Trace>:MODE`

class ModeCls

Mode commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → TraceModeC

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
value: enums.TraceModeC = driver.display.window.trace.mode.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

mode: No help available

set(*mode: TraceModeC, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE
driver.display.window.trace.mode.set(mode = enums.TraceModeC.AVERage, window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.mode.clone()
```

Subgroups

6.6.16.9.2.1 Hcontinuous

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:MODE:HCONtinuous
```

class HcontinuousCls

Hcontinuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE:HCONtinuous
value: bool = driver.display.window.trace.mode.hcontinuous.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

hold: No help available

set(hold: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:MODE:HCONtinuous
driver.display.window.trace.mode.hcontinuous.set(hold = False, window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param hold

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.3 Smoothing

class SmoothingCls

Smoothing commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.smoothing.clone()
```

Subgroups

6.6.16.9.3.1 Aperture

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SMOothing:APERture
```

class ApertureCls

Aperture commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing:APERture
value: float = driver.display.window.trace.smoothing.aperture.get(window = ↵
↵repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the magnitude (aperture) of trace smoothing.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on trace smoothing (method RsFswp.Display.Window.Trace.Smoothing.State.set) .

In the Spot Noise vs Tune measurement, trace smoothing applies to either all traces or none. Use [SENSe:]SMOothing[:STATe] in that case.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

ref_value: No help available

set(ref_value: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing:APERture
driver.display.window.trace.smoothing.aperture.set(ref_value = 1.0, window = ↵
↵repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the magnitude (aperture) of trace smoothing.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on trace smoothing (method RsFswp.Display.Window.Trace.Smoothing.State.set) .

In the Spot Noise vs Tune measurement, trace smoothing applies to either all traces or none. Use [SENSe:]SMOothing[:STATe] in that case.

param ref_value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.3.2 State

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:SMOothing:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing[:STATe]
value: bool = driver.display.window.trace.smoothing.state.get(window = repcap.
↳ Window.Default, trace = repcap.Trace.Default)
```

This command turns trace smoothing for a specific trace on and off. When you turn on trace smoothing, you can define the smoothing magnitude with method RsFswp.Display.Window.Trace.Smoothing.Aperture.set. In the Spot Noise vs Tune measurement, trace smoothing applies to either all traces or none. Use [SENSe:]SMOothing[:STATe] in that case.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: ON | OFF | 1 | 0

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:SMOothing[:STATe]
driver.display.window.trace.smoothing.state.set(state = False, window = repcap.
↳ Window.Default, trace = repcap.Trace.Default)
```

This command turns trace smoothing for a specific trace on and off. When you turn on trace smoothing, you can define the smoothing magnitude with method `RsFswp.Display.Window.Trace.Smoothing.Aperture.set`. In the Spot Noise vs Tune measurement, trace smoothing applies to either all traces or none. Use `[SENSe:]SMOothing[:STATe]` in that case.

param state

ON | OFF | 1 | 0

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.4 State

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*, trace=*Trace.Default*) → bool

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
value: bool = driver.display.window.trace.state.get(window = repcap.Window.
↳Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

state: No help available

set(state: bool, window=*Window.Default*, trace=*Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>[:STATe]
driver.display.window.trace.state.set(state = False, window = repcap.Window.
↳Default, trace = repcap.Trace.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.5 Y**class YCls**

Y commands group definition. 11 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.y.clone()
```

Subgroups**6.6.16.9.5.1 Scale****SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe
```

class ScaleCls

Scale commands group definition. 10 total commands, 8 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]
value: float = driver.display.window.trace.y.scale.get(window = repcap.Window.
↳Default, trace = repcap.Trace.Default)
```

This command defines the value range displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic scaling of the y-axis (method RsFswp.Display.Window.Trace.Y.Scale.Auto.set)

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

scale: No help available

set(scale: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]
driver.display.window.trace.y.scale.set(scale = 1.0, window = repcap.Window.
↳Default, trace = repcap.Trace.Default)
```

This command defines the value range displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic scaling of the y-axis (method RsFswp.Display.Window.Trace.Y.Scale.Auto.set)

param scale

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.y.scale.clone()
```

Subgroups

6.6.16.9.5.2 Auto

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(auto: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:AUTO
driver.display.window.trace.y.scale.auto.set(auto = False, window = repcap.
↳ Window.Default, trace = repcap.Trace.Default)
```

This command turns automatic scaling of the y-axis in graphical result displays on and off.

param auto

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.9.5.3 Maximum

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MAXimum
value: float = driver.display.window.trace.y.scale.maximum.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

max_py: No help available

set(*max_py: float, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MAXimum
driver.display.window.trace.y.scale.maximum.set(max_py = 1.0, window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param max_py

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.5.4 Minimum

SCPI Commands

`DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:MINimum`

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MINimum
value: float = driver.display.window.trace.y.scale.minimum.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

min_py: No help available

set(*min_py: float, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:MINimum
driver.display.window.trace.y.scale.minimum.set(min_py = 1.0, window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

No command help available

param min_py

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.5.5 Mode

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → ReferenceMode

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:MODE
value: enums.ReferenceMode = driver.display.window.trace.y.scale.mode.
↪get(window = repcap.Window.Default, trace = repcap.Trace.Default)
```

This command selects how frequencies are displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Frequency result display must be available and selected.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

reference_mode: No help available

set(reference_mode: ReferenceMode, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:MODE
driver.display.window.trace.y.scale.mode.set(reference_mode = enums.
↪ReferenceMode.Absolute, window = repcap.Window.Default, trace = repcap.Trace.
↪Default)
```

This command selects how frequencies are displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Frequency result display must be available and selected.

param reference_mode

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.16.9.5.6 Pdivision

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:PDIVision
```

class PdivisionCls

Pdivision commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:PDIVision
value: float = driver.display.window.trace.y.scale.pdivision.get(window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the range of a single diagram division on the y-axis of the phase result display.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

per_division: No help available

set(per_division: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:PDIVision
driver.display.window.trace.y.scale.pdivision.set(per_division = 1.0, window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the range of a single diagram division on the y-axis of the phase result display.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param per_division

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.9.5.7 RefLevel

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel
value: float = driver.display.window.trace.y.scale.refLevel.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

This command defines the maximum level displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic scaling of the y-axis (method RsFswp.Display.Window.Trace.Y.Scale.Auto.set)
-

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

value: No help available

set(value: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel
driver.display.window.trace.y.scale.refLevel.set(value = 1.0, window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

This command defines the maximum level displayed on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic scaling of the y-axis (method RsFswp.Display.Window.Trace.Y.Scale.Auto.set)
-

param value

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.display.window.trace.y.scale.refLevel.clone()
```

Subgroups

6.6.16.9.5.8 Offset

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RLEVel:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet
value: float = driver.display.window.trace.y.scale.refLevel.offset.get(window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the amount by which a trace is shifted.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on trace offset (DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet:STATe) .

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

offset: numeric value Unit: dB

set(offset: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet
driver.display.window.trace.y.scale.refLevel.offset.set(offset = 1.0, window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the amount by which a trace is shifted.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on trace offset (DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RLEVel:OFFSet:STATe) .

param offset

numeric value Unit: dB

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.9.5.9 RefPosition**SCPI Commands**

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALE:RPOStion
```

class RefPositionCls

RefPosition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:RPOStion
value: float = driver.display.window.trace.y.scale.refPosition.get(window = ↵
↵repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the position of the reference value on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

position: Percentage of the diagram height with 100 % corresponding to the upper diagram border Unit: PCT

set(position: float, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALE]:RPOStion
driver.display.window.trace.y.scale.refPosition.set(position = 1.0, window = ↵
↵repcap.Window.Default, trace = repcap.Trace.Default)
```

This command defines the position of the reference value on the y-axis.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param position

Percentage of the diagram height with 100 % corresponding to the upper diagram border Unit: PCT

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.9.5.10 Rvalue

SCPI Commands

`DISPlay:WINDow<Window>:TRACe<Trace>:Y:SCALe:RVALue`

class RvalueCls

Rvalue commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default, trace=Trace.Default*) → float

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RVALue
value: float = driver.display.window.trace.y.scale.rvalue.get(window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

This command defines the value assigned to the reference position.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

return

ref_value: numeric value Unit: Depends on the selected unit (deg or rad)

set(*ref_value: float, window=Window.Default, trace=Trace.Default*) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y[:SCALe]:RVALue
driver.display.window.trace.y.scale.rvalue.set(ref_value = 1.0, window = repcap.
↳Window.Default, trace = repcap.Trace.Default)
```

This command defines the value assigned to the reference position.

INTRO_CMD_HELP: Prerequisites for this command

- Phase result display must be available and selected.

param ref_value

numeric value Unit: Depends on the selected unit (deg or rad)

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Window’)

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface ‘Trace’)

6.6.16.9.5.11 Spacing

SCPI Commands

```
DISPlay:WINDow<Window>:TRACe<Trace>:Y:SPACing
```

class SpacingCls

Spacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → SpacingY

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y:SPACing
value: enums.SpacingY = driver.display.window.trace.y.spacing.get(window =
↳repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

spacing: No help available

set(spacing: SpacingY, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: DISPlay[:WINDow<n>]:TRACe<t>:Y:SPACing
driver.display.window.trace.y.spacing.set(spacing = enums.SpacingY.LDB, window
↳= repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param spacing

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.6.17 Wselect

SCPI Commands

DISPlay:WSElect

class WselectCls

Wselect commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: DISPlay:WSElect
value: int = driver.display.wselect.get()
```

This command queries the currently active window (the one that is focused) in the currently selected measurement channel.

return

selected_window: Index number of the currently active window. Range: 1 to 16

6.7 Fetch

class FetchCls

Fetch commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.fetch.clone()
```

Subgroups

6.7.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.fetch.pmeter.repcap_powerMeter_get()
driver.fetch.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```


SCPI Commands

```
FETCh:PMETer<PowerMeter>
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

get(powerMeter=PowerMeter.Default) → List[float]

```
# SCPI: FETCh:PMETer<p>
value: List[float] = driver.fetch.pmeter.get(powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.fetch.pmeter.clone()
```

6.8 FormatPy

class FormatPyCls

FormatPy commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.formatPy.clone()
```

Subgroups

6.8.1 Data

SCPI Commands

```
FORMat:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DataFormat

```
# SCPI: FORMat[:DATA]
value: enums.DataFormat = driver.formatPy.data.get()
```

This command selects the data format that is used for transmission of trace data from the R&S FSWP to the controlling computer. Note that the command has no effect for data that you send to the R&S FSWP. The R&S FSWP automatically recognizes the data it receives, regardless of the format. For details on data formats, see ‘Formats for returned values: ASCII format and binary format’.

return
data_format: No help available

set(data_format: DataFormat) → None

```
# SCPI: FORMat[:DATA]
driver.formatPy.data.set(data_format = enums.DataFormat.ASCII)
```

This command selects the data format that is used for transmission of trace data from the R&S FSWP to the controlling computer. Note that the command has no effect for data that you send to the R&S FSWP. The R&S FSWP automatically recognizes the data it receives, regardless of the format. For details on data formats, see ‘Formats for returned values: ASCII format and binary format’.

param data_format
No help available

6.8.2 Dexport

class DexportCls

Dexport commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.formatPy.dexport.clone()
```

Subgroups

6.8.2.1 Dseparator

SCPI Commands

```
FORMat:DEXPort:DSEPARATOR
```

class DseparatorCls

Dseparator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Separator

```
# SCPI: FORMat:DEXPort:DSEPARATOR
value: enums.Separator = driver.formatPy.dexport.dseparator.get()
```

This command selects the decimal separator for data exported in ASCII format.

return

separator: POINT | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05.

POINTt Uses a point as decimal separator, e.g. 4.05.

set(separator: Separator) → None

```
# SCPI: FORMat:DEXPort:DSEParator
driver.formatPy.dexport.dseparator.set(separator = enums.Separator.COMMa)
```

This command selects the decimal separator for data exported in ASCII format.

param separator

POINTt | COMMa COMMa Uses a comma as decimal separator, e.g. 4,05. POINTt Uses

a point as decimal separator, e.g. 4.05.

6.8.2.2 Header

SCPI Commands

```
FORMat:DEXPort:HEADer
```

class HeaderCls

Header commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: FORMat:DEXPort:HEADer
value: bool = driver.formatPy.dexport.header.get()
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: FORMat:DEXPort:HEADer
driver.formatPy.dexport.header.set(state = False)
```

If enabled, additional instrument and measurement settings are included in the header of the export file for result data. If disabled, only the pure result data from the selected traces and tables is exported.

param state

ON | OFF | 0 | 1

6.8.2.3 Traces

SCPI Commands

FORMat:DEXPort:TRACes

class TracesCls

Traces commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SelectionScope

```
# SCPI: FORMat:DEXPort:TRACes
value: enums.SelectionScope = driver.formatPy.dexport.traces.get()
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set) .

return

selection: SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

set(selection: SelectionScope) → None

```
# SCPI: FORMat:DEXPort:TRACes
driver.formatPy.dexport.traces.set(selection = enums.SelectionScope.ALL)
```

This command selects the data to be included in a data export file (see method RsFswp.MassMemory.Store.Trace.set) .

param selection

SINGLE | ALL SINGLE Only a single trace is selected for export, namely the one specified by the method RsFswp.MassMemory.Store.Trace.set command. ALL Selects all active traces and result tables (e.g. 'Result Summary', marker peak list etc.) in the current application for export to an ASCII file. The trace parameter for the method RsFswp.MassMemory.Store.Trace.set command is ignored.

6.8.2.4 Xdistrib

SCPI Commands

FORMat:DEXPort:XDIStrib

class XdistribCls

Xdistrib commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Xdistribution

```
# SCPI: FORMat:DEXPort:XDIStrib
value: enums.Xdistribution = driver.formatPy.dexport.xdistrib.get()
```

No command help available

return

xdistribution: No help available

set(xdistribution: *Xdistribution*) → None

```
# SCPI: FORMat:DEXPort:XDIStrib
driver.formatPy.dexport.xdistrib.set(xdistribution = enums.Xdistribution.
↳BINCentered)
```

No command help available

param xdistribution

No help available

6.9 HardCopy

SCPI Commands

HCOpy:ABORt

class HardCopyCls

HardCopy commands group definition. 60 total commands, 15 Subgroups, 1 group commands

abort() → None

```
# SCPI: HCOpy:ABORt
driver.hardCopy.abort()
```

This command aborts a running hardcopy output.

abort_with_opc(opc_timeout_ms: *int = -1*) → None

```
# SCPI: HCOpy:ABORt
driver.hardCopy.abort_with_opc()
```

This command aborts a running hardcopy output.

Same as abort, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.clone()
```

Subgroups

6.9.1 Cmap<Item>

RepCap Settings

```
# Range: Ix1 .. Ix64
rc = driver.hardCopy.cmap.repcap_item_get()
driver.hardCopy.cmap.repcap_item_set(repcap.Item.Ix1)
```

class CmapCls

Cmap commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: Item, default value after init: Item.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.cmap.clone()
```

Subgroups

6.9.1.1 Default<Colors>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.hardCopy.cmap.default.repcap_colors_get()
driver.hardCopy.cmap.default.repcap_colors_set(repcap.Colors.Ix1)
```

SCPI Commands

```
HCOPY:CMAP<Item>:DEFAult<Colors>
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Colors, default value after init: Colors.Ix1

set(*item*=Item.Default, *colors*=Colors.Default) → None

```
# SCPI: HCOpy:CMAP<it>:DEFAult<ci>
driver.hardCopy.cmap.default.set(item = repcap.Item.Default, colors = repcap.
↪ Colors.Default)
```

This command defines the color scheme for print jobs. For details see ‘Print Colors’.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Cmap’)

param colors

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Default')

set_with_opc(*item=Item.Default, colors=Colors.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.cmap.default.clone()
```

6.9.1.2 Hsl**SCPI Commands**

```
HCOPY:CMAP<Item>:HSL
```

class HslCls

Hsl commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class HslStruct

Response structure. Fields:

- Hue: float: hue tint Range: 0 to 1
- Sat: float: sat saturation Range: 0 to 1
- Lum: float: lum brightness Range: 0 to 1

get(*item=Item.Default*) → HslStruct

```
# SCPI: HCOpy:CMAP<it>:HSL
value: HslStruct = driver.hardCopy.cmap.hsl.get(item = repcap.Item.Default)
```

This command selects the color for various screen elements in print jobs.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

return

structure: for return value, see the help for HslStruct structure arguments.

set(*hue: float, sat: float, lum: float, item=Item.Default*) → None

```
# SCPI: HCOpy:CMAP<it>:HSL
driver.hardCopy.cmap.hsl.set(hue = 1.0, sat = 1.0, lum = 1.0, item = repcap.
↪Item.Default)
```

This command selects the color for various screen elements in print jobs.

param hue

hue tint Range: 0 to 1

param sat

sat saturation Range: 0 to 1

param lum

lum brightness Range: 0 to 1

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

6.9.1.3 Pdefined

SCPI Commands

HCOpy:CMAP<Item>:PDEFined

class PdefinedCls

Pdefined commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*item=Item.Default*) → Color

```
# SCPI: HCOpy:CMAP<it>:PDEFined
value: enums.Color = driver.hardCopy.cmap.pdefined.get(item = repcap.Item.
↳Default)
```

This command selects a predefined color for various screen elements in print jobs.

param item

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

returncolor: BLACK | BLUE | BROWn | GREen | CYAN | RED | MAGenta | YELLow |
WHITe | DGRay | LGRay | LBLue | LGReen | LCYan | LRED | LMAGenta**set**(*color: Color, item=Item.Default*) → None

```
# SCPI: HCOpy:CMAP<it>:PDEFined
driver.hardCopy.cmap.pdefined.set(color = enums.Color.BLACK, item = repcap.Item.
↳Default)
```

This command selects a predefined color for various screen elements in print jobs.

param colorBLACK | BLUE | BROWn | GREen | CYAN | RED | MAGenta | YELLow | WHITe |
DGRay | LGRay | LBLue | LGReen | LCYan | LRED | LMAGenta**param item**

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Cmap')

6.9.2 Comment

SCPI Commands

HCOPy:COMMeNt

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOPy:COMMeNt
value: str = driver.hardCopy.comment.get()
```

No command help available

return
comment: No help available

set(comment: str) → None

```
# SCPI: HCOPy:COMMeNt
driver.hardCopy.comment.set(comment = '1')
```

No command help available

param comment
No help available

6.9.3 Content

SCPI Commands

HCOPy:CONTent

class ContentCls

Content commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → HardcopyContent

```
# SCPI: HCOPy:CONTent
value: enums.HardcopyContent = driver.hardCopy.content.get()
```

This command determines the type of content included in the printout. This setting is independent of the printing device.

return
content: WINDows | HCOPy WINDows Includes only the selected windows in the printout. All currently active windows for the current channel (or 'MultiView') are available for selection. How many windows are printed on a each page of the printout is defined by method RsFswp.HardCopy.Page.Window.Count.set. This option is not available when copying to the clipboard (HCOP:DEST 'SYST:COMM:CLIP' or an image file (see method RsFswp.HardCopy.Device.Language.set) . If the destination is currently set to an image file or the clipboard, it is automatically changed to be a PDF file for the currently selected printing device. HCOPy Selects all measurement

results displayed on the screen for the current channel (or 'MultiView') : diagrams, traces, markers, marker lists, limit lines, etc., including the channel bar and status bar, for printout on a single page. Displayed items belonging to the software user interface (e.g. softkeys) are not included. The size and position of the elements in the printout is identical to the screen display.

set(*content: HardcopyContent*) → None

```
# SCPI: HCOpy:CONTent
driver.hardCopy.content.set(content = enums.HardcopyContent.HCOpy)
```

This command determines the type of content included in the printout. This setting is independent of the printing device.

param content

WINDows | HCOpy WINDows Includes only the selected windows in the printout. All currently active windows for the current channel (or 'MultiView') are available for selection. How many windows are printed on a each page of the printout is defined by method RsFswp.HardCopy.Page.Window.Count.set. This option is not available when copying to the clipboard (HCOP:DEST 'SYST:COMM:CLIP' or an image file (see method RsFswp.HardCopy.Device.Language.set) . If the destination is currently set to an image file or the clipboard, it is automatically changed to be a PDF file for the currently selected printing device. HCOpy Selects all measurement results displayed on the screen for the current channel (or 'MultiView') : diagrams, traces, markers, marker lists, limit lines, etc., including the channel bar and status bar, for printout on a single page. Displayed items belonging to the software user interface (e.g. softkeys) are not included. The size and position of the elements in the printout is identical to the screen display.

6.9.4 Copies

SCPI Commands

```
HCOPy:COPIes
```

class CopiesCls

Copies commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOpy:COPIes
value: float = driver.hardCopy.copies.get()
```

No command help available

return

copies: No help available

set(*copies: float*) → None

```
# SCPI: HCOpy:COPIes
driver.hardCopy.copies.set(copies = 1.0)
```

No command help available

param copies
No help available

6.9.5 Destination

SCPI Commands

HCOPy:DESTination

class DestinationCls

Destination commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:DESTination
value: str = driver.hardCopy.destination.get()
```

This command selects the destination of a print job. Note: To print a screenshot to a file, see method RsFswp.HardCopy.Device.Language.set.

return

destination: 'MMEM' Activates 'Print to file'. Thus, if the destination of the print function is set to 'printer' (see HCOP:DEST1 'SYSTem:COMMunicate:PRINter' or HCOP:DEV:LANG GDI) , the output is redirected to a .PRN file using the selected printer driver. Select the file name with method RsFswp.MassMemory.Name.set. Note: To save a screenshot to a file, see method RsFswp.HardCopy.Device.Language.set. 'SYSTem:COMMunicate:PRINter' Sends the hardcopy to a printer and deactivates 'print to file'. Select the printer with SYSTem:COMMunicate:PRINter:SElectdi . 'SYSTem:COMMunicate:CLIPboard' Sends the hardcopy to the clipboard.

set(destination: str) → None

```
# SCPI: HCOpy:DESTination
driver.hardCopy.destination.set(destination = '1')
```

This command selects the destination of a print job. Note: To print a screenshot to a file, see method RsFswp.HardCopy.Device.Language.set.

param destination

'MMEM' Activates 'Print to file'. Thus, if the destination of the print function is set to 'printer' (see HCOP:DEST1 'SYSTem:COMMunicate:PRINter' or HCOP:DEV:LANG GDI) , the output is redirected to a .PRN file using the selected printer driver. Select the file name with method RsFswp.MassMemory.Name.set. Note: To save a screenshot to a file, see method RsFswp.HardCopy.Device.Language.set. 'SYSTem:COMMunicate:PRINter' Sends the hardcopy to a printer and deactivates 'print to file'. Select the printer with SYSTem:COMMunicate:PRINter:SElectdi . 'SYSTem:COMMunicate:CLIPboard' Sends the hardcopy to the clipboard.

6.9.6 Device

class DeviceCls

Device commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.device.clone()
```

Subgroups

6.9.6.1 Color

SCPI Commands

```
HCOPY:DEVICE:COLor
```

class ColorCls

Color commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:DEvice:COLor
value: bool = driver.hardCopy.device.color.get()
```

This command turns color printing on and off.

return

state: ON | OFF | 0 | 1 ON | 1 Color printing OFF | 0 Black and white printing

set(state: bool) → None

```
# SCPI: HCOpy:DEvice:COLor
driver.hardCopy.device.color.set(state = False)
```

This command turns color printing on and off.

param state

ON | OFF | 0 | 1 ON | 1 Color printing OFF | 0 Black and white printing

6.9.6.2 Language

SCPI Commands

```
HCOPY:DEVICE:LANGUage
```

class LanguageCls

Language commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → *PictureFormat*

```
# SCPI: HCOpy:DEvice:LANGuage
value: enums.PictureFormat = driver.hardCopy.device.language.get()
```

This command selects the file format for a print job or to store a screenshot to a file.

return

language: GDI Graphics Device Interface Default format for output to a printer configured under Windows. Must be selected for output to the printer interface. Can be used for output to a file. The printer driver configured under Windows is used to generate a printer-specific file format. BMP | JPG | PNG | PDF | SVG Data format for output to files

set(*language: PictureFormat*) → *None*

```
# SCPI: HCOpy:DEvice:LANGuage
driver.hardCopy.device.language.set(language = enums.PictureFormat.BMP)
```

This command selects the file format for a print job or to store a screenshot to a file.

param language

GDI Graphics Device Interface Default format for output to a printer configured under Windows. Must be selected for output to the printer interface. Can be used for output to a file. The printer driver configured under Windows is used to generate a printer-specific file format. BMP | JPG | PNG | PDF | SVG Data format for output to files

6.9.7 Immediate

SCPI Commands

HCOpy:IMMediate

class ImmediateCls

Immediate commands group definition. 2 total commands, 1 Subgroups, 1 group commands

set() → *None*

```
# SCPI: HCOpy[:IMMediate]
driver.hardCopy.immediate.set()
```

This command initiates a print job. If you are printing to a file, the file name depends on method RsFswp.MassMemory.Name. set.

set_with_opc(*opc_timeout_ms: int = -1*) → *None*

```
# SCPI: HCOpy[:IMMediate]
driver.hardCopy.immediate.set_with_opc()
```

This command initiates a print job. If you are printing to a file, the file name depends on method RsFswp.MassMemory.Name. set.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.immediate.clone()
```

Subgroups**6.9.7.1 Next****SCPI Commands**

```
HCOPY:IMMediate:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy[:IMMediate]:NEXT
driver.hardCopy.immediate.next.set()
```

This command initiates a print job. If you are printing to a file, the file name depends on method RsFswp.MassMemory.Name. set. This command adds a consecutive number to the file name.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy[:IMMediate]:NEXT
driver.hardCopy.immediate.next.set_with_opc()
```

This command initiates a print job. If you are printing to a file, the file name depends on method RsFswp.MassMemory.Name. set. This command adds a consecutive number to the file name.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.8 Item**class ItemCls**

Item commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.item.clone()
```

Subgroups

6.9.8.1 All

SCPI Commands

```
HCOPY:ITEM:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOPY:ITEM:ALL
driver.hardCopy.item.all.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOPY:ITEM:ALL
driver.hardCopy.item.all.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.8.2 Window

class WindowCls

Window commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.item.window.clone()
```

Subgroups

6.9.8.2.1 Text

SCPI Commands

```
HCOPY:ITEM:WINDow:TEXT
```

class TextCls

Text commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:ITEM:WINDow:TEXT
value: str = driver.hardCopy.item.window.text.get()
```

This command defines a comment to be added to the printout.

return
comment: String containing the comment.

set(comment: str) → None

```
# SCPI: HCOpy:ITEM:WINDow:TEXT
driver.hardCopy.item.window.text.set(comment = '1')
```

This command defines a comment to be added to the printout.

param comment
String containing the comment.

6.9.8.2.2 Trace

class TraceCls

Trace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.item.window.trace.clone()
```

Subgroups

6.9.8.2.2.1 State

SCPI Commands

```
HCOPY:ITEM:WINDow:TRACe:STATe
```


class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:ITEM:WINDow:TRACe:STATe
value: bool = driver.hardCopy.item.window.trace.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: HCOpy:ITEM:WINDow:TRACe:STATe
driver.hardCopy.item.window.trace.state.set(state = False)
```

No command help available

```
param state
    No help available
```

6.9.9 Logo

SCPI Commands

HCOpy:LOGO

class LogoCls

Logo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:LOGO
value: bool = driver.hardCopy.logo.get()
```

No command help available

```
return
    filename: No help available
```

set(filename: bool) → None

```
# SCPI: HCOpy:LOGO
driver.hardCopy.logo.set(filename = False)
```

No command help available

```
param filename
    No help available
```

6.9.10 Mode

SCPI Commands

`HCOPY:MODE`

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → HardcopyMode

```
# SCPI: HCOpy:MODE
value: enums.HardcopyMode = driver.hardCopy.mode.get()
```

No command help available

return
mode: No help available

set(mode: HardcopyMode) → None

```
# SCPI: HCOpy:MODE
driver.hardCopy.mode.set(mode = enums.HardcopyMode.REPort)
```

No command help available

param mode
No help available

6.9.11 Page

class PageCls

Page commands group definition. 11 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.page.clone()
```

Subgroups

6.9.11.1 Count

class CountCls

Count commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.page.count.clone()
```

Subgroups

6.9.11.1.1 State

SCPI Commands

```
HCOPY:PAGE:COUNt:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:PAGE:COUNt:STATe
value: bool = driver.hardCopy.page.count.state.get()
```

This command includes or excludes the page number for printouts consisting of multiple pages (method RsFswp.HardCopy. Content.set) .

return

state: 1 | 0 | ON | OFF 1 | ON The page number is printed. 0 | OFF The page number is not printed.

set(state: bool) → None

```
# SCPI: HCOpy:PAGE:COUNt:STATe
driver.hardCopy.page.count.state.set(state = False)
```

This command includes or excludes the page number for printouts consisting of multiple pages (method RsFswp.HardCopy. Content.set) .

param state

1 | 0 | ON | OFF 1 | ON The page number is printed. 0 | OFF The page number is not printed.

6.9.11.2 Margin

class MarginCls

Margin commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.page.margin.clone()
```

Subgroups

6.9.11.2.1 Bottom

SCPI Commands

```
HCOPy:PAGE:MARGIn:BOTTom
```

class BottomCls

Bottom commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOPy:PAGE:MARGIn:BOTTom
value: float = driver.hardCopy.page.margin.bottom.get()
```

This command defines the margin at the bottom of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

return

bottom: No help available

set(bottom: float) → None

```
# SCPI: HCOPy:PAGE:MARGIn:BOTTom
driver.hardCopy.page.margin.bottom.set(bottom = 1.0)
```

This command defines the margin at the bottom of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

param bottom

No help available

6.9.11.2.2 Left

SCPI Commands

```
HCOPy:PAGE:MARGIn:LEFT
```

class LeftCls

Left commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOPy:PAGE:MARGIn:LEFT
value: float = driver.hardCopy.page.margin.left.get()
```

This command defines the margin at the left side of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

return
left: No help available

set(*left: float*) → None

```
# SCPI: HCOpy:PAGE:MARGin:LEFT
driver.hardCopy.page.margin.left.set(left = 1.0)
```

This command defines the margin at the left side of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

param left
No help available

6.9.11.2.3 Right

SCPI Commands

```
HCOpy:PAGE:MARGin:RIGHT
```

class RightCls

Right commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOpy:PAGE:MARGin:RIGHT
value: float = driver.hardCopy.page.margin.right.get()
```

This command defines the margin at the right side of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

return
right: No help available

set(*right: float*) → None

```
# SCPI: HCOpy:PAGE:MARGin:RIGHT
driver.hardCopy.page.margin.right.set(right = 1.0)
```

This command defines the margin at the right side of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

param right
No help available

6.9.11.2.4 Top

SCPI Commands

HCOpy:PAGE:MARGin:TOP

class TopCls

Top commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOpy:PAGE:MARGin:TOP
value: float = driver.hardCopy.page.margin.top.get()
```

This command defines the margin at the top of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

return
top: No help available

set(top: float) → None

```
# SCPI: HCOpy:PAGE:MARGin:TOP
driver.hardCopy.page.margin.top.set(top = 1.0)
```

This command defines the margin at the top of the printout page on which no elements are printed. The margins are defined according to method RsFswp.HardCopy.Page.Margin.Unit.set.

param top
No help available

6.9.11.2.5 Unit

SCPI Commands

HCOpy:PAGE:MARGin:UNIT

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PageMarginUnit

```
# SCPI: HCOpy:PAGE:MARGin:UNIT
value: enums.PageMarginUnit = driver.hardCopy.page.margin.unit.get()
```

This command defines the unit in which the margins for the printout page are configured.

return
unit: MM | IN MM millimeters IN inches

set(unit: PageMarginUnit) → None

```
# SCPI: HCOpy:PAGE:MARGin:UNIT
driver.hardCopy.page.margin.unit.set(unit = enums.PageMarginUnit.IN)
```

This command defines the unit in which the margins for the printout page are configured.

param unit
MM | IN MM millimeters IN inches

6.9.11.3 Orientation

SCPI Commands

```
HCOPY:PAGE:ORIENTATION
```

class OrientationCls

Orientation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PageOrientation

```
# SCPI: HCOpy:PAGE:ORIENTATION
value: enums.PageOrientation = driver.hardCopy.page.orientation.get()
```

The command selects the page orientation of the printout. The command is only available if the output device is a printer or a PDF file.

return
orientation: LANDscape | PORTrait

set(orientation: PageOrientation) → None

```
# SCPI: HCOpy:PAGE:ORIENTATION
driver.hardCopy.page.orientation.set(orientation = enums.PageOrientation.
↳LANDscape)
```

The command selects the page orientation of the printout. The command is only available if the output device is a printer or a PDF file.

param orientation
LANDscape | PORTrait

6.9.11.4 Window

class WindowCls

Window commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.page.window.clone()
```

Subgroups

6.9.11.4.1 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.page.window.channel.clone()
```

Subgroups

6.9.11.4.1.1 State

SCPI Commands

```
HCOPY:PAGE:WINDow:CHANnel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class StateStruct

Response structure. Fields:

- Channel: str: String containing the name of the channel. For a list of available channel types use [CMDLINK: INSTRument:LIST? CMDLINK].
- State: bool: 1 | 0 | ON | OFF 1 | ON The channel windows are included in the printout. 0 | OFF The channel windows are not included in the printout.

get() → StateStruct

```
# SCPI: HCOpy:PAGE:WINDow:CHANnel:STATe
value: StateStruct = driver.hardCopy.page.window.channel.state.get()
```

This command selects all windows of the specified channel to be included in the printout for method RsFswp.HardCopy.Content.set.

return

structure: for return value, see the help for StateStruct structure arguments.

set(channel: str, state: bool) → None

```
# SCPI: HCOpy:PAGE:WINDow:CHANnel:STATe
driver.hardCopy.page.window.channel.state.set(channel = '1', state = False)
```

This command selects all windows of the specified channel to be included in the printout for method RsFswp.HardCopy.Content.set.

param channel

String containing the name of the channel. For a list of available channel types use method **Rsfswp.Instrument.ListPy.get_**.

param state

1 | 0 | ON | OFF 1 | ON The channel windows are included in the printout. 0 | OFF The channel windows are not included in the printout.

6.9.11.4.2 Count**SCPI Commands**

HCOpy:PAGE:WINDow:COUNT

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: HCOpy:PAGE:WINDow:COUNT
value: float = driver.hardCopy.page.window.count.get()
```

This command defines how many windows are displayed on a single page of the printout for method RsFswp.HardCopy.Content. set.

return

count: integer

set(count: float) → None

```
# SCPI: HCOpy:PAGE:WINDow:COUNT
driver.hardCopy.page.window.count.set(count = 1.0)
```

This command defines how many windows are displayed on a single page of the printout for method RsFswp.HardCopy.Content. set.

param count

integer

6.9.11.4.3 Scale**SCPI Commands**

HCOpy:PAGE:WINDow:SCALE

class ScaleCls

Scale commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:PAGE:WINDow:SCALE
value: bool = driver.hardCopy.page.window.scale.get()
```

This command determines the scaling of the windows in the printout for method RsFswp.HardCopy.Content.set.

return

scale: 1 | 0 | ON | OFF 1 | ON Each window is scaled to fit the page size optimally, not regarding the aspect ratio of the original display. If more than one window is printed on one page (see method RsFswp.HardCopy.Page.Window.Count.set), each window is printed in equal size. ('Size to fit') 0 | OFF Each window is printed as large as possible while maintaining the aspect ratio of the original display. ('Maintain aspect ratio')

set(scale: bool) → None

```
# SCPI: HCOpy:PAGE:WINDow:SCALE
driver.hardCopy.page.window.scale.set(scale = False)
```

This command determines the scaling of the windows in the printout for method RsFswp.HardCopy.Content.set.

param scale

1 | 0 | ON | OFF 1 | ON Each window is scaled to fit the page size optimally, not regarding the aspect ratio of the original display. If more than one window is printed on one page (see method RsFswp.HardCopy.Page.Window.Count.set), each window is printed in equal size. ('Size to fit') 0 | OFF Each window is printed as large as possible while maintaining the aspect ratio of the original display. ('Maintain aspect ratio')

6.9.11.4.4 State

SCPI Commands

```
HCOpy:PAGE:WINDow:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class StateStruct

Response structure. Fields:

- Channel: str: String containing the name of the channel. For a list of available channel types use [CMDLINK: INSTRument:LIST? CMDLINK].
- Window: str: String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the [CMDLINK: LAYout:CATalog[:WINDow]? CMDLINK] query.
- State: bool: 1 | 0 | ON | OFF 1 | ON The window is included in the printout. 0 | OFF The window is not included in the printout.

get() → StateStruct

```
# SCPI: HCOpy:PAGE:WINDow:STATe
value: StateStruct = driver.hardCopy.page.window.state.get()
```

This command selects the windows to be included in the printout for method RsFswp.HardCopy.Content.set.

return

structure: for return value, see the help for StateStruct structure arguments.

set(channel: str, window: str, state: bool) → None

```
# SCPI: HCOpy:PAGE:WINDow:STATe
driver.hardCopy.page.window.state.set(channel = '1', window = '1', state = False)
```

This command selects the windows to be included in the printout for method RsFswp.HardCopy.Content.set.

param channel

String containing the name of the channel. For a list of available channel types use method **RsFswp.Instrument.ListPy.get_**.

param window

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method **RsFswp.Layout.Catalog.Window.get_** query.

param state

1 | 0 | ON | OFF 1 | ON The window is included in the printout. 0 | OFF The window is not included in the printout.

6.9.12 Print

SCPI Commands

HCOpy:PRINt

class PrintCls

Print commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:PRINt
driver.hardCopy.print.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:PRINt
driver.hardCopy.print.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.13 TdDtamp

class TdDtampCls

TdDtamp commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.tdDtamp.clone()
```

Subgroups

6.9.13.1 State

SCPI Commands

```
HCOPY:TDSTamp:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TDSTamp:STATE
value: bool = driver.hardCopy.tdDtamp.state.get()
```

This command includes or excludes the time and date in the printout.

return

state: 1 | 0 | ON | OFF 1 | ON The time and date are printed. 0 | OFF The time and date are not printed.

set(state: bool) → None

```
# SCPI: HCOpy:TDSTamp:STATE
driver.hardCopy.tdDtamp.state.set(state = False)
```

This command includes or excludes the time and date in the printout.

param state

1 | 0 | ON | OFF 1 | ON The time and date are printed. 0 | OFF The time and date are not printed.

6.9.14 Theme

class ThemeCls

Theme commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.theme.clone()
```

Subgroups

6.9.14.1 Select

SCPI Commands

```
HCOPY:THEME:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*theme: str*) → None

```
# SCPI: HCOPY:THEME:SElect
driver.hardCopy.theme.select.set(theme = '1')
```

No command help available

param theme
No help available

6.9.15 Treport

class TreportCls

Treport commands group definition. 29 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.clone()
```

Subgroups

6.9.15.1 Append

SCPI Commands

HCOpy:TREPort:APPend

class AppendCls

Append commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:APPend
driver.hardCopy.treport.append.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:APPend
driver.hardCopy.treport.append.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.2 Description

SCPI Commands

HCOpy:TREPort:DESCription

class DescriptionCls

Description commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:TREPort:DESCription
value: str = driver.hardCopy.treport.description.get()
```

No command help available

return

description: No help available

set(description: str) → None

```
# SCPI: HCOpy:TREPort:DESCription
driver.hardCopy.treport.description.set(description = '1')
```

No command help available

param description

No help available

6.9.15.3 Item

class ItemCls

Item commands group definition. 13 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.item.clone()
```

Subgroups

6.9.15.3.1 Default

SCPI Commands

```
HCOPY:TREPort:ITEM:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:ITEM:DEFault
driver.hardCopy.treport.item.default.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:ITEM:DEFault
driver.hardCopy.treport.item.default.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.3.2 Header

class HeaderCls

Header commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.item.header.clone()
```

Subgroups

6.9.15.3.2.1 Line<Line>

RepCap Settings

```
# Range: Ix1 .. Ix8
rc = driver.hardCopy.treport.item.header.line.repcap_line_get()
driver.hardCopy.treport.item.header.line.repcap_line_set(repcap.Line.Ix1)
```

class LineCls

Line commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability:
Line, default value after init: Line.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.item.header.line.clone()
```

Subgroups

6.9.15.3.2.2 Control

SCPI Commands

```
HCOPY:TREPort:ITEM:HEADer:LINE<Line>:CONTRol
```

class ControlCls

Control commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(line=Line.Default) → HardcopyHeader

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:CONTRol
value: enums.HardcopyHeader = driver.hardCopy.treport.item.header.line.control.
↪get(line = repcap.Line.Default)
```

No command help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

return

repetition: No help available

set(*repetition: HardcopyHeader*, *line=Line.Default*) → None

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:CONtrol
driver.hardCopy.treport.item.header.line.control.set(repetition = enums.
↳HardcopyHeader.ALways, line = repcap.Line.Default)
```

No command help available

param repetition

No help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

6.9.15.3.2.3 Text**SCPI Commands**

```
HCOpy:TREPort:ITEM:HEADer:LINE<Line>:TEXT
```

class TextCls

Text commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*line=Line.Default*) → str

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:TEXT
value: str = driver.hardCopy.treport.item.header.line.text.get(line = repcap.
↳Line.Default)
```

No command help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

return

description: No help available

set(*description: str*, *line=Line.Default*) → None

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:TEXT
driver.hardCopy.treport.item.header.line.text.set(description = '1', line =
↳repcap.Line.Default)
```

No command help available

param description

No help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

6.9.15.3.2.4 Title**SCPI Commands**

HCOpy:TREPort:ITEM:HEADer:LINE<Line>:TITLE

class TitleCls

Title commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*line=Line.Default*) → str

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:TITLE
value: str = driver.hardCopy.treport.item.header.line.title.get(line = repcap.
↳Line.Default)
```

No command help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

return

title: No help available

set(*title: str, line=Line.Default*) → None

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:LINE<li>:TITLE
driver.hardCopy.treport.item.header.line.title.set(title = '1', line = repcap.
↳Line.Default)
```

No command help available

param title

No help available

param line

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Line')

6.9.15.3.2.5 State**SCPI Commands**

HCOpy:TREPort:ITEM:HEADer:STAtE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:STAtE
value: bool = driver.hardCopy.treport.item.header.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: HCOpy:TREPort:ITEM:HEADer:STAtE
driver.hardCopy.treport.item.header.state.set(state = False)
```

No command help available

```
param state
    No help available
```

6.9.15.3.3 ListPy

SCPI Commands

```
HCOpy:TREPort:ITEM:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:TREPort:ITEM:LIST
value: str = driver.hardCopy.treport.item.listPy.get()
```

No command help available

```
return
    channel_type: No help available
```

6.9.15.3.4 Logo

SCPI Commands

```
HCOpy:TREPort:ITEM:LOGO
```

class LogoCls

Logo commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:TREPort:ITEM:LOGO
value: str = driver.hardCopy.treport.item.logo.get()
```

No command help available

return

filename: No help available

set(filename: str) → None

```
# SCPI: HCOpy:TREPort:ITEM:LOGO
driver.hardCopy.treport.item.logo.set(filename = '1')
```

No command help available

param filename

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.item.logo.clone()
```

Subgroups

6.9.15.3.4.1 Control

SCPI Commands

```
HCOPy:TREPort:ITEM:LOGO:CONTRol
```

class ControlCls

Control commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → HardcopyLogo

```
# SCPI: HCOpy:TREPort:ITEM:LOGO:CONTRol
value: enums.HardcopyLogo = driver.hardCopy.treport.item.logo.control.get()
```

No command help available

return

repetition: No help available

set(repetition: HardcopyLogo) → None

```
# SCPI: HCOpy:TREPort:ITEM:LOGO:CONTRol
driver.hardCopy.treport.item.logo.control.set(repetition = enums.HardcopyLogo.
↪ALWays)
```

No command help available

param repetition

No help available

6.9.15.3.5 Select

SCPI Commands

```
HCOPy:TREPort:ITEM:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*item: str, arg_1: Optional[str] = None*) → None

```
# SCPI: HCOPy:TREPort:ITEM:SElect
driver.hardCopy.treport.item.select.set(item = r1, arg_1 = '1')
```

No command help available

param item

No help available

param arg_1

No help available

6.9.15.3.6 Template

SCPI Commands

```
HCOPy:TREPort:ITEM:TEMPlate:DElete
HCOPy:TREPort:ITEM:TEMPlate:LOAD
HCOPy:TREPort:ITEM:TEMPlate:SAVE
```

class TemplateCls

Template commands group definition. 4 total commands, 1 Subgroups, 3 group commands

delete(*template: str*) → None

```
# SCPI: HCOPy:TREPort:ITEM:TEMPlate:DElete
driver.hardCopy.treport.item.template.delete(template = '1')
```

No command help available

param template

No help available

load(*template: str*) → None

```
# SCPI: HCOPy:TREPort:ITEM:TEMPlate:LOAD
driver.hardCopy.treport.item.template.load(template = '1')
```

No command help available

param template

No help available

save(*template: str*) → None

```
# SCPI: HCOpy:TREPort:ITEM:TEMPlate:SAVE
driver.hardCopy.treport.item.template.save(template = '1')
```

No command help available

param template

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.item.template.clone()
```

Subgroups

6.9.15.3.6.1 Catalog

SCPI Commands

```
HCOpy:TREPort:ITEM:TEMPlate:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: HCOpy:TREPort:ITEM:TEMPlate:CATalog
value: List[str] = driver.hardCopy.treport.item.template.catalog.get()
```

No command help available

return

result: No help available

6.9.15.4 New

SCPI Commands

```
HCOpy:TREPort:NEW
```

class NewCls

New commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:NEW
driver.hardCopy.treport.new.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:NEW
driver.hardCopy.treport.new.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.5 Pagecount

class PagecountCls

Pagecount commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.pagecount.clone()
```

Subgroups

6.9.15.5.1 State

SCPI Commands

```
HCOpy:TREPort:PAGecount:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TREPort:PAGecount:STATE
value: bool = driver.hardCopy.treport.pagecount.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: HCOpy:TREPort:PAGecount:STATE
driver.hardCopy.treport.pagecount.state.set(state = False)
```

No command help available

param state

No help available

6.9.15.6 PageSize

SCPI Commands

`HCOPY:TREPort:PAgeSize`

class PageSizeCls

PageSize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → HardcopyPageSize

```
# SCPI: HCOpy:TREPort:PAgeSize
value: enums.HardcopyPageSize = driver.hardCopy.treport.pageSize.get()
```

No command help available

return

size: No help available

set(size: HardcopyPageSize) → None

```
# SCPI: HCOpy:TREPort:PAgeSize
driver.hardCopy.treport.pageSize.set(size = enums.HardcopyPageSize.A4)
```

No command help available

param size

No help available

6.9.15.7 Pcolors

class PcolorsCls

Pcolors commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.pcolors.clone()
```

Subgroups

6.9.15.7.1 State

SCPI Commands

`HCOPY:TREPort:PCOLors:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TREPort:PCOLors:STATE
value: bool = driver.hardCopy.treport.pcolors.state.get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: HCOpy:TREPort:PCOLors:STATE
driver.hardCopy.treport.pcolors.state.set(state = False)
```

No command help available

```
param state
No help available
```

6.9.15.8 TdDtamp

class TdDtampCls

TdDtamp commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.tdDtamp.clone()
```

Subgroups

6.9.15.8.1 State

SCPI Commands

```
HCOpy:TREPort:TDSTamp:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TREPort:TDSTamp:STATE
value: bool = driver.hardCopy.treport.tdDtamp.state.get()
```

No command help available

```
return
state: No help available
```

set(*state: bool*) → None

```
# SCPI: HCOpy:TREPort:TDStamp:STATE
driver.hardCopy.treport.tdDtamp.state.set(state = False)
```

No command help available

param state

No help available

6.9.15.9 Test

class TestCls

Test commands group definition. 7 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.test.clone()
```

Subgroups

6.9.15.9.1 Remove

SCPI Commands

```
HCOpy:TREPort:TEST:REMove
```

class RemoveCls

Remove commands group definition. 3 total commands, 2 Subgroups, 1 group commands

set(*data_set: float*) → None

```
# SCPI: HCOpy:TREPort:TEST:REMove
driver.hardCopy.treport.test.remove.set(data_set = 1.0)
```

No command help available

param data_set

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.test.remove.clone()
```

Subgroups

6.9.15.9.1.1 All

SCPI Commands

```
HCOPY:TREPort:TEST:REMove:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:TEST:REMove:ALL
driver.hardCopy.treport.test.remove.all.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:TEST:REMove:ALL
driver.hardCopy.treport.test.remove.all.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.9.1.2 Selected

SCPI Commands

```
HCOPY:TREPort:TEST:REMove:SElected
```

class SelectedCls

Selected commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:TEST:REMove:SElected
driver.hardCopy.treport.test.remove.selected.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:TEST:REMove:SElected
driver.hardCopy.treport.test.remove.selected.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.9.2 Select

SCPI Commands

`HCOPY:TREPort:TEST:SElect`

class SelectCls

Select commands group definition. 4 total commands, 3 Subgroups, 1 group commands

set(selection: float, state: bool) → None

```
# SCPI: HCOpy:TREPort:TEST:SElect
driver.hardCopy.treport.test.select.set(selection = 1.0, state = False)
```

No command help available

param selection

No help available

param state

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.test.select.clone()
```

Subgroups

6.9.15.9.2.1 All

SCPI Commands

`HCOPY:TREPort:TEST:SElect:ALL`

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:ALL
driver.hardCopy.treport.test.select.all.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:ALL
driver.hardCopy.treport.test.select.all.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.9.2.2 Invert

SCPI Commands

```
HCOpy:TREPort:TEST:SElect:INVert
```

class InvertCls

Invert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:INVert
driver.hardCopy.treport.test.select.invert.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:INVert
driver.hardCopy.treport.test.select.invert.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.9.2.3 NonePy

SCPI Commands

```
HCOpy:TREPort:TEST:SElect:NONE
```

class NonePyCls

NonePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:NONE
driver.hardCopy.treport.test.select.nonePy.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: HCOpy:TREPort:TEST:SElect:NONE
driver.hardCopy.treport.test.select.nonePy.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.9.15.10 Title

SCPI Commands

```
HCOpy:TREPort:TITLe
```

class TitleCls

Title commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → str

```
# SCPI: HCOpy:TREPort:TITLe
value: str = driver.hardCopy.treport.title.get()
```

No command help available

return

title: No help available

set(title: str) → None

```
# SCPI: HCOpy:TREPort:TITLe
driver.hardCopy.treport.title.set(title = '1')
```

No command help available

param title

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.hardCopy.treport.title.clone()
```

Subgroups

6.9.15.10.1 State

SCPI Commands

```
HCOPY:TREPort:TITLe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: HCOpy:TREPort:TITLe:STATe
value: bool = driver.hardCopy.treport.title.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: HCOpy:TREPort:TITLe:STATe
driver.hardCopy.treport.title.state.set(state = False)
```

No command help available

param state

No help available

6.10 Initiate

SCPI Commands

```
INITiate:IMMediate
```

class InitiateCls

Initiate commands group definition. 12 total commands, 6 Subgroups, 1 group commands

immediate() → None

```
# SCPI: INITiate[:IMMediate]
driver.initiate.immediate()
```

This command starts a (single) new measurement. With measurement count or average count > 0, this means a restart of the corresponding number of measurements. With trace mode MAXHold, MINHold and AVERage, the previous results are reset on restarting the measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

immediate_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate[:IMMediate]
driver.initiate.immediate_with_opc()
```

This command starts a (single) new measurement. With measurement count or average count > 0, this means a restart of the corresponding number of measurements. With trace mode MAXHold, MINHold and AVERage, the previous results are reset on restarting the measurement. You can synchronize to the end of the measurement with **OPC*, **OPC?* or **WAI*. For details on synchronization see Remote control via SCPI.

Same as immediate, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.initiate.clone()
```

Subgroups

6.10.1 Block

SCPI Commands

```
INITiate:BLOCK:ABORt
```

class BlockCls

Block commands group definition. 3 total commands, 2 Subgroups, 1 group commands

abort() → None

```
# SCPI: INITiate:BLOCK:ABORt
driver.initiate.block.abort()
```

No command help available

abort_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:BLOCK:ABORt
driver.initiate.block.abort_with_opc()
```


No command help available

Same as abort, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.initiate.block.clone()
```

Subgroups

6.10.1.1 ConMeas

SCPI Commands

```
INITiate:BLOCK:CONMeas
```

class ConMeasCls

ConMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(group_name: Optional[str] = None) → None

```
# SCPI: INITiate:BLOCK:CONMeas
driver.initiate.block.conMeas.set(group_name = '1')
```

No command help available

param group_name

No help available

6.10.1.2 Immediate

SCPI Commands

```
INITiate:BLOCK:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(group_name: Optional[str] = None) → None

```
# SCPI: INITiate:BLOCK:IMMediate
driver.initiate.block.immediate.set(group_name = '1')
```

No command help available

param group_name

No help available

6.10.2 ConMeas

SCPI Commands

INITiate:CONMeas

class ConMeasCls

ConMeas commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:CONMeas
driver.initiate.conMeas.set()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:CONMeas
driver.initiate.conMeas.set_with_opc()
```

This command restarts a (single) measurement that has been stopped (using method RsFswp.#Abort CMDLINKRESOLVED]) or finished in single measurement mode. The measurement is restarted at the beginning, not where the previous measurement was stopped. As opposed to [CMDLINKRESOLVED Applications.K30_NoiseFigure.Initiate.Immediate.set, this command does not reset traces in maxhold, minhold or average mode. Therefore it can be used to continue measurements using maxhold or averaging functions.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.10.3 Continuous

SCPI Commands

INITiate:CONTinuous

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: INITiate:CONTinuous
driver.initiate.continuous.set(state = False)
```

This command controls the measurement mode for an individual channel. Note that in single measurement mode, you can synchronize to the end of the measurement with ***OPC**, ***OPC?** or ***WAI**. In continuous measurement mode, synchronization to the end of the measurement is not possible. Thus, it is not recommended that you use continuous measurement mode in remote control, as results like trace data or markers are only valid after a single measurement end synchronization. If the measurement mode is changed for a channel while the Sequencer is active the mode is only considered the next time the measurement in that channel is activated by the Sequencer.

param state

ON | OFF | 0 | 1 ON | 1 Continuous measurement OFF | 0 Single measurement

6.10.4 Spectrum

SCPI Commands

```
INITiate:ESpectrum
```

class SpectrumCls

Spectrum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:ESpectrum
driver.initiate.spectrum.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:ESpectrum
driver.initiate.spectrum.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.10.5 Sequencer

SCPI Commands

```
INITiate:SEQuencer:ABORt
```

class SequencerCls

Sequencer commands group definition. 4 total commands, 3 Subgroups, 1 group commands

abort() → None

```
# SCPI: INITiate:SEQuencer:ABORt
driver.initiate.sequencer.abort()
```

This command stops the currently active sequence of measurements. You can start a new sequence any time using method RsFswp.Initiate.Sequencer.Immediate.set.

abort_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SEQuencer:ABORt
driver.initiate.sequencer.abort_with_opc()
```

This command stops the currently active sequence of measurements. You can start a new sequence any time using method RsFswp.Initiate.Sequencer.Immediate.set.

Same as abort, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.initiate.sequencer.clone()
```

Subgroups

6.10.5.1 Immediate

SCPI Commands

```
INITiate:SEQuencer:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:SEQuencer:IMMediate
driver.initiate.sequencer.immediate.set()
```

This command starts a new sequence of measurements by the Sequencer. Its effect is similar to the method RsFswp.Applications.K30_NoiseFigure.Initiate.Immediate.set command used for a single measurement. Before this command can be executed, the Sequencer must be activated (see method RsFswp.System.Sequencer.set) .

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SEQuencer:IMMediate
driver.initiate.sequencer.immediate.set_with_opc()
```

This command starts a new sequence of measurements by the Sequencer. Its effect is similar to the method RsFswp.Applications.K30_NoiseFigure.Initiate.Immediate.set command used for a single measurement. Before this command can be executed, the Sequencer must be activated (see method RsFswp.System.Sequencer.set) .

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.10.5.2 Mode**SCPI Commands**

```
INITiate:SEQuencer:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(mode: SequencerMode) → None

```
# SCPI: INITiate:SEQuencer:MODE
driver.initiate.sequencer.mode.set(mode = enums.SequencerMode.CDEFINED)
```

Defines the capture mode for the entire measurement sequence and all measurement groups and channels it contains. Note: To synchronize to the end of a measurement sequence using *OPC, *OPC? or *WAI, use SINGLE Sequencer mode.

param mode

SINGLE Each measurement group is started one after the other in the order of definition. All measurement channels in a group are started simultaneously and performed once. After all measurements are completed, the next group is started. After the last group, the measurement sequence is finished. **CONTInuous** Each measurement group is started one after the other in the order of definition. All measurement channels in a group are started simultaneously and performed once. After all measurements are completed, the next group is started. After the last group, the measurement sequence restarts with the first one and continues until it is stopped explicitly.

6.10.5.3 Refresh**class RefreshCls**

Refresh commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.initiate.sequencer.refresh.clone()
```

Subgroups**6.10.5.3.1 All****SCPI Commands**

```
INITiate:SEQuencer:REFresh:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:SEQuencer:REFresh[:ALL]
driver.initiate.sequencer.refresh.all.set()
```

This function is only available if the Sequencer is deactivated (SYST:SEQ:OFF) and only in MSRA mode. The data in the capture buffer is re-evaluated by all active MSRA secondary applications.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SEQuencer:REFresh[:ALL]
driver.initiate.sequencer.refresh.all.set_with_opc()
```

This function is only available if the Sequencer is deactivated (SYST:SEQ:OFF) and only in MSRA mode. The data in the capture buffer is re-evaluated by all active MSRA secondary applications.

Same as set, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.10.6 Spurious

SCPI Commands

```
INITiate:SPURious
```

class SpuriousCls

Spurious commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: INITiate:SPURious
driver.initiate.spurious.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INITiate:SPURious
driver.initiate.spurious.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.11 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.inputPy.repcap_inputIx_get()
driver.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 46 total commands, 17 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.clone()
```

Subgroups

6.11.1 Attenuation

SCPI Commands

```
INPut:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:ATTenuation
value: float = driver.inputPy.attenuation.get()
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

return

attenuation: Range: see data sheet , Unit: DB

set(attenuation: float) → None

```
# SCPI: INPut:ATTenuation
driver.inputPy.attenuation.set(attenuation = 1.0)
```

This command defines the total attenuation for RF input. If you set the attenuation manually, it is no longer coupled to the reference level, but the reference level is coupled to the attenuation. Thus, if the current reference level is not compatible with an attenuation that has been set manually, the command also adjusts the reference level.

param attenuation

Range: see data sheet , Unit: DB

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.attenuation.clone()
```

Subgroups**6.11.1.1 Auto****SCPI Commands**

```
INPut:ATTenuation:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: INPut:ATTenuation:AUTO
driver.inputPy.attenuation.auto.set(state = False)
```

This command couples or decouples the attenuation to the reference level. Thus, when the reference level is changed, the R&S FSWP determines the signal level for optimal internal data processing and sets the required attenuation accordingly.

param state

ON | OFF | 0 | 1

6.11.1.2 Protection**SCPI Commands**

```
INPut<InputIx>:ATTenuation:PROTection:RESet
```

class ProtectionCls

Protection commands group definition. 1 total commands, 0 Subgroups, 1 group commands

reset(device_name: Optional[str] = None, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:ATTenuation:PROTection:RESet
driver.inputPy.attenuation.protection.reset(device_name = '1', inputIx = repcap.
↪InputIx.Default)
```

This command resets the attenuator and reconnects the RF input with the input mixer for the R&S FSWP after an overload condition occurred and the protection mechanism intervened. The error status bit (bit 3 in the method RsFswp.Status. **Questionable.Power.Event.get_** status register) and the INPUT OVLD message in the status bar are cleared. (For details on the status register see the R&S FSWP base unit user manual) . The command works only if the overload condition has been eliminated first.

param device_name

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.2 Connector

SCPI Commands

INPut:CONNector

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(connector: InputConnectorB) → None

```
# SCPI: INPut:CONNector
driver.inputPy.connector.set(connector = enums.InputConnectorB.AIQI)
```

This command selects the measurement channel for baseband noise measurements.

param connector

No help available

6.11.3 Coupling

SCPI Commands

INPut:COUpling

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CouplingTypeA

```
# SCPI: INPut:COUpling
value: enums.CouplingTypeA = driver.inputPy.coupling.get()
```

This command selects the coupling type of the RF input.

return

coupling_type: AC | DC AC AC coupling DC DC coupling

set(coupling_type: CouplingTypeA) → None

```
# SCPI: INPut:COUpling
driver.inputPy.coupling.set(coupling_type = enums.CouplingTypeA.AC)
```

This command selects the coupling type of the RF input.

param coupling_type

AC | DC AC AC coupling DC DC coupling

6.11.4 Diq

class DiqCls

Diq commands group definition. 7 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.diq.clone()
```

Subgroups

6.11.4.1 Cdevice

SCPI Commands

```
INPut:DIQ:CDEvice
```

class CdeviceCls

Cdevice commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: INPut:DIQ:CDEvice
value: str = driver.inputPy.diq.cdevice.get()
```

No command help available

return
result: No help available

6.11.4.2 Range

class RangeCls

Range commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.diq.range.clone()
```

Subgroups

6.11.4.2.1 Coupling

SCPI Commands

```
INPut:DIQ:RANGe:COUPling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:DIQ:RANGe:COUPling
value: bool = driver.inputPy.diq.range.coupling.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: bool) → None

```
# SCPI: INPut:DIQ:RANGe:COUPling
driver.inputPy.diq.range.coupling.set(arg_0 = False)
```

No command help available

```
param arg_0
    No help available
```

6.11.4.2.2 Upper

SCPI Commands

```
INPut:DIQ:RANGe:UPPer
```

class UpperCls

Upper commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:DIQ:RANGe[:UPPer]
value: float = driver.inputPy.diq.range.upper.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: float) → None

```
# SCPI: INPut:DIQ:RANGe[:UPPer]
driver.inputPy.diq.range.upper.set(arg_0 = 1.0)
```

No command help available

param arg_0

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.diq.range.upper.clone()
```

Subgroups

6.11.4.2.2.1 Auto

SCPI Commands

```
INPut:DIQ:RANGe:UPPer:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: INPut:DIQ:RANGe[:UPPer]:AUTO
driver.inputPy.diq.range.upper.auto.set(state = False)
```

No command help available

param state

No help available

6.11.4.2.2.2 Unit

SCPI Commands

```
INPut:DIQ:RANGe:UPPer:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → DiqUnit

```
# SCPI: INPut:DIQ:RANGe[:UPPer]:UNIT
value: enums.DiqUnit = driver.inputPy.diq.range.upper.unit.get()
```

No command help available

return

arg_0: No help available

set(arg_0: *DiqUnit*) → None

```
# SCPI: INPut:DIQ:RANGe[:UPPer]:UNIT
driver.inputPy.diq.range.upper.unit.set(arg_0 = enums.DiqUnit.AMPere)
```

No command help available

param arg_0
No help available

6.11.4.3 SymbolRate

SCPI Commands

INPut:DIQ:SRATe

class SymbolRateCls

SymbolRate commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:DIQ:SRATe
value: float = driver.inputPy.diq.symbolRate.get()
```

No command help available

return
arg_0: No help available

set(arg_0: float) → None

```
# SCPI: INPut:DIQ:SRATe
driver.inputPy.diq.symbolRate.set(arg_0 = 1.0)
```

No command help available

param arg_0
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.diq.symbolRate.clone()
```

Subgroups

6.11.4.3.1 Auto

SCPI Commands

INPut:DIQ:SRATe:AUTO

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*state: bool*) → None

```
# SCPI: INPut:DIQ:SRATe:AUTO
driver.inputPy.diq.symbolRate.auto.set(state = False)
```

No command help available

param state

No help available

6.11.5 Dpath

SCPI Commands

INPut:DPATH

class DpathCls

Dpath commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoOrOff

```
# SCPI: INPut:DPATH
value: enums.AutoOrOff = driver.inputPy.dpath.get()
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

return

direct_path_setting: No help available

set(*direct_path_setting: AutoOrOff*) → None

```
# SCPI: INPut:DPATH
driver.inputPy.dpath.set(direct_path_setting = enums.AutoOrOff.AUTO)
```

Enables or disables the use of the direct path for frequencies close to 0 Hz.

param direct_path_setting

No help available

6.11.6 Eatt

SCPI Commands

```
INPut:EATT
```

class EattCls

Eatt commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:EATT
value: float = driver.inputPy.eatt.get()
```

No command help available

return

attenuation: No help available

set(attenuation: float) → None

```
# SCPI: INPut:EATT
driver.inputPy.eatt.set(attenuation = 1.0)
```

No command help available

param attenuation

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.eatt.clone()
```

Subgroups

6.11.6.1 Auto

SCPI Commands

```
INPut:EATT:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: INPut:EATT:AUTO
driver.inputPy.eatt.auto.set(state = False)
```

No command help available

param state

No help available

6.11.6.2 State

SCPI Commands

```
INPut:EATT:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:EATT:STATe
value: bool = driver.inputPy.eatt.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: INPut:EATT:STATe
driver.inputPy.eatt.state.set(state = False)
```

No command help available

param state
No help available

6.11.7 Egain

class EgainCls

Egain commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.egain.clone()
```

Subgroups

6.11.7.1 State

SCPI Commands

```
INPut<InputIx>:EGain:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → bool

```
# SCPI: INPut<ip>:EGain[:STATE]
value: bool = driver.inputPy.egain.state.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: No help available

set(*state: bool, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:EGain[:STATE]
driver.inputPy.egain.state.set(state = False, inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.8 File

class FileCls

File commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.file.clone()
```

Subgroups

6.11.8.1 Path

SCPI Commands

```
INPut:FILE:PATH
```

class PathCls

Path commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: INPut:FILE:PATH
value: str = driver.inputPy.file.path.get()
```

This command selects the I/Q data file to be used as input for further measurements. The I/Q data must have a specific format as described in ‘I/Q data file format (iq-tar)’. For details, see ‘Basics on input from I/Q data files’.

return

file_path: No help available

set(file_path: str, analysis_bw: float) → None

```
# SCPI: INPut:FILE:PATH
driver.inputPy.file.path.set(file_path = '1', analysis_bw = 1.0)
```

This command selects the I/Q data file to be used as input for further measurements. The I/Q data must have a specific format as described in ‘I/Q data file format (iq-tar)’. For details, see ‘Basics on input from I/Q data files’.

param file_path

No help available

param analysis_bw

Optionally: The analysis bandwidth to be used by the measurement. The bandwidth must be smaller than or equal to the bandwidth of the data that was stored in the file.

Unit: HZ

6.11.8.2 Zpadding

SCPI Commands

```
INPut<InputIx>:FILE:ZPADing
```

class ZpaddingCls

Zpadding commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<1|2>:FILE:ZPADing
value: bool = driver.inputPy.file.zpadding.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

arg_0: No help available

set(arg_0: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<1|2>:FILE:ZPADing
driver.inputPy.file.zpadding.set(arg_0 = False, inputIx = repcap.InputIx.Default)
```

No command help available

param arg_0

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.9 FilterPy

class FilterPyCls

FilterPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.filterPy.clone()
```

Subgroups

6.11.9.1 Hpass

class HpassCls

Hpass commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.filterPy.hpass.clone()
```

Subgroups

6.11.9.1.1 State

SCPI Commands

```
INPut:FILTer:HPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTer:HPASs[:STATe]
value: bool = driver.inputPy.filterPy.hpass.state.get()
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-presselector, if available.)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: INPut:FILTer:HPASs[:STATe]
driver.inputPy.filterPy.hpass.state.set(state = False)
```

Activates an additional internal high-pass filter for RF input signals from 1 GHz to 3 GHz. This filter is used to remove the harmonics of the R&S FSWP to measure the harmonics for a DUT, for example. This function requires an additional high-pass filter hardware option. (Note: for RF input signals outside the specified range, the high-pass filter has no effect. For signals with a frequency of approximately 4 GHz upwards, the harmonics are suppressed sufficiently by the YIG-preselector, if available.)

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.11.9.2 Yig

class YigCls

Yig commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.filterPy.yig.clone()
```

Subgroups

6.11.9.2.1 State

SCPI Commands

```
INPut:FILTer:YIG:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:FILTer:YIG[:STATe]
value: bool = driver.inputPy.filterPy.yig.state.get()
```

Enables or disables the YIG filter.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: INPut:FILTer:YIG[:STATe]
driver.inputPy.filterPy.yig.state.set(state = False)
```

Enables or disables the YIG filter.

param state
ON | OFF | 0 | 1

6.11.10 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.gain.clone()
```

Subgroups

6.11.10.1 State

SCPI Commands

```
INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:GAIN:STATe
value: bool = driver.inputPy.gain.state.get()
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

return
state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: INPut:GAIN:STATe
driver.inputPy.gain.state.set(state = False)
```

This command turns the internal preamplifier on and off. It requires the optional preamplifier hardware. The preamplification value is defined using the method RsFswp.Applications.K30_NoiseFigure.InputPy.Gain.Value.set.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.11.10.2 Value**SCPI Commands**

INPut:GAIN:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:GAIN[:VALue]
value: float = driver.inputPy.gain.value.get()
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications. K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

return

level: No help available

set(level: float) → None

```
# SCPI: INPut:GAIN[:VALue]
driver.inputPy.gain.value.set(level = 1.0)
```

This command selects the ‘gain’ if the preamplifier is activated (INP:GAIN:STAT ON, see method RsFswp.Applications. K30_NoiseFigure.InputPy.Gain.State.set) . The command requires the additional preamplifier hardware option.

param level

No help available

6.11.11 Impedance**SCPI Commands**

INPut:IMPedance

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: INPut:IMPedance
value: int = driver.inputPy.impedance.get()
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

return

impedance: 50|75 numeric value User-defined impedance from 50 Ohm to 100000000 Ohm (=100 MOhm) User-defined values are only available for the Spectrum application, the I/Q Analyzer, and some optional applications. Unit: OHM

set(impedance: int) → None

```
# SCPI: INPut:IMPedance
driver.inputPy.impedance.set(impedance = 1)
```

This command selects the nominal input impedance of the RF input. In some applications, only 50 are supported.

param impedance

50|75 numeric value User-defined impedance from 50 Ohm to 100000000 Ohm (=100 MOhm) User-defined values are only available for the Spectrum application, the I/Q Analyzer, and some optional applications. Unit: OHM

6.11.12 Iq

class IqCls

Iq commands group definition. 14 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.clone()
```

Subgroups

6.11.12.1 Balanced

class BalancedCls

Balanced commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.balanced.clone()
```

Subgroups

6.11.12.1.1 State

SCPI Commands

INPut:IQ:BALEnced:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INPut:IQ:BALEnced:STATe
value: bool = driver.inputPy.iq.balanced.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: INPut:IQ:BALEnced:STATe
driver.inputPy.iq.balanced.state.set(state = False)
```

No command help available

param state

No help available

6.11.12.2 Fullscale

class FullscaleCls

Fullscale commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.fullscale.clone()
```

Subgroups

6.11.12.2.1 Auto

SCPI Commands

INPut:IQ:FULLscale:AUTO

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*state: bool*) → None

```
# SCPI: INPut:IQ:FULLscale:AUTO
driver.inputPy.iq.fullscale.auto.set(state = False)
```

No command help available

param state
No help available

6.11.12.2.2 Level**SCPI Commands**

```
INPut:IQ:FULLscale:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: INPut:IQ:FULLscale:LEVel
value: float = driver.inputPy.iq.fullscale.level.get()
```

No command help available

return
value: No help available

set(*value: float*) → None

```
# SCPI: INPut:IQ:FULLscale:LEVel
driver.inputPy.iq.fullscale.level.set(value = 1.0)
```

No command help available

param value
No help available

6.11.12.3 Osc**class OscCls**

Osc commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.clone()
```

Subgroups

6.11.12.3.1 Balanced

class BalancedCls

Balanced commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.balanced.clone()
```

Subgroups

6.11.12.3.1.1 State

SCPI Commands

```
INPut<InputIx>:IQ:OSC:BAnced:StAte
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:IQ:OSC:BAnced[:StAte]
value: bool = driver.inputPy.iq.osc.balanced.state.get(inputIx = repcap.InputIx.
↳Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:BAnced[:StAte]
driver.inputPy.iq.osc.balanced.state.set(state = False, inputIx = repcap.
↳InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.12.3.2 Fullscale**class FullscaleCls**

Fullscale commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.fullscale.clone()
```

Subgroups**6.11.12.3.2.1 Level****SCPI Commands**

```
INPut<InputIx>:IQ:OSC:FULLscale:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:FULLscale[:LEVel]
value: float = driver.inputPy.iq.osc.fullscale.level.get(inputIx = repcap.
↳InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

level: No help available

set(level: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:FULLscale[:LEVel]
driver.inputPy.iq.osc.fullscale.level.set(level = 1.0, inputIx = repcap.InputIx.
↳Default)
```

No command help available

param level

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.12.3.3 Skew**class SkewCls**

Skew commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.skew.clone()
```

Subgroups**6.11.12.3.3.1 Icomponent****SCPI Commands**

```
INPut<InputIx>:IQ:OSC:SKEW:I
```

class IcomponentCls

Icomponent commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I
value: float = driver.inputPy.iq.osc.skew.icomponent.get(inputIx = repcap.
↪InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I
driver.inputPy.iq.osc.skew.icomponent.set(value = 1.0, inputIx = repcap.InputIx.
↪Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.skew.icomponent.clone()
```

Subgroups**6.11.12.3.3.2 Inverted****SCPI Commands**

```
INPut<InputIx>:IQ:OSC:SKEW:I:INVerted
```

class InvertedCls

Inverted commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I:INVerted
value: float = driver.inputPy.iq.osc.skew.icomponent.inverted.get(inputIx = ↵
↵repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:I:INVerted
driver.inputPy.iq.osc.skew.icomponent.inverted.set(value = 1.0, inputIx = ↵
↵repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.12.3.3.3 Qcomponent

SCPI Commands

`INPut<InputIx>:IQ:OSC:SKEW:Q`

class QcomponentCls

Qcomponent commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q
value: float = driver.inputPy.iq.osc.skew.qcomponent.get(inputIx = repcap.
↳InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

value: No help available

set(*value: float, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q
driver.inputPy.iq.osc.skew.qcomponent.set(value = 1.0, inputIx = repcap.InputIx.
↳Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.iq.osc.skew.qcomponent.clone()
```

Subgroups

6.11.12.3.3.4 Inverted

SCPI Commands

`INPut<InputIx>:IQ:OSC:SKEW:Q:INVerted`

class InvertedCls

Inverted commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q:INVerted
value: float = driver.inputPy.iq.osc.skew.qcomponent.inverted.get(inputIx =
↳repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

value: No help available

set(value: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:IQ:OSC:SKEW:Q:INVerted
driver.inputPy.iq.osc.skew.qcomponent.inverted.set(value = 1.0, inputIx =
↳repcap.InputIx.Default)
```

No command help available

param value

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.12.3.4 State**SCPI Commands**

```
INPut<InputIx>:IQ:OSC:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:IQ:OSC[:STATe]
value: bool = driver.inputPy.iq.osc.state.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(*state: bool, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC[:STATe]
driver.inputPy.iq.osc.state.set(state = False, inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.12.3.5 SymbolRate

SCPI Commands

```
INPut<InputIx>:IQ:OSC:SRATe
```

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → float

```
# SCPI: INPut<ip>:IQ:OSC:SRATe
value: float = driver.inputPy.iq.osc.symbolRate.get(inputIx = repcap.InputIx.
↪Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

sample_rate: No help available

set(*sample_rate: float, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:SRATe
driver.inputPy.iq.osc.symbolRate.set(sample_rate = 1.0, inputIx = repcap.
↪InputIx.Default)
```

No command help available

param sample_rate

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.12.3.6 TcpiP

SCPI Commands

```
INPut<InputIx>:IQ:OSC:TcPiP
```

class TcpiPcls

TcpiP commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → str

```
# SCPI: INPut<ip>:IQ:OSC:TcPiP
value: str = driver.inputPy.iq.osc.tcpiP.get(inputIx = rePcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

tcpiP: No help available

set(*tcpiP: str, inputIx=InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:TcPiP
driver.inputPy.iq.osc.tcpiP.set(tcpiP = '1', inputIx = rePcap.InputIx.Default)
```

No command help available

param tcpiP

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.12.3.7 TypePy

SCPI Commands

```
INPut<InputIx>:IQ:OSC:TYPE
```

class TypePycls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → IqType

```
# SCPI: INPut<ip>:IQ:OSC:TYPE
value: enums.IqType = driver.inputPy.iq.osc.typePy.get(inputIx = rePcap.InputIx.
↪Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

type_py: No help available

set(type_py: *IqType*, inputIx=*InputIx.Default*) → None

```
# SCPI: INPut<ip>:IQ:OSC:TYPE
driver.inputPy.iq.osc.typePy.set(type_py = enums.IqType.Ipart=I, inputIx = ↵
↵repcap.InputIx.Default)
```

No command help available

param type_py

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.12.4 TypePy

SCPI Commands

```
INPut:IQ:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → *IqType*

```
# SCPI: INPut:IQ:TYPE
value: enums.IqType = driver.inputPy.iq.typePy.get()
```

No command help available

return

iq_type: No help available

set(iq_type: *IqType*) → None

```
# SCPI: INPut:IQ:TYPE
driver.inputPy.iq.typePy.set(iq_type = enums.IqType.Ipart=I)
```

No command help available

param iq_type

No help available

6.11.13 Loscillator

class LoscillatorCls

Loscillator commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.loscillator.clone()
```

Subgroups

6.11.13.1 Source

SCPI Commands

```
INPut<InputIx>:LOSCillator:SOURce
```

class SourceCls

Source commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → SourceInt

```
# SCPI: INPut<ip>:LOSCillator:SOURce
value: enums.SourceInt = driver.inputPy.loscillator.source.get(inputIx = repcap.
↳InputIx.Default)
```

This command selects the type of local oscillator in the test setup.

INTRO_CMD_HELP: Prerequisites for this command

- Select additive noise or pulsed additive noise measurement (CONFigure:PNOise:MEASurement)
-

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

location: EXTernal External local oscillator connected to the 'LO AUX Input' of the R&S FSWP. INTernal Internal local oscillator of the R&S FSWP.

set(location: SourceInt, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:LOSCillator:SOURce
driver.inputPy.loscillator.source.set(location = enums.SourceInt.EXTernal,
↳inputIx = repcap.InputIx.Default)
```

This command selects the type of local oscillator in the test setup.

INTRO_CMD_HELP: Prerequisites for this command

- Select additive noise or pulsed additive noise measurement (CONFigure:PNOise:MEASurement)
-

param location

EXTernal External local oscillator connected to the ‘LO AUX Input’ of the R&S FSWP.
INTernal Internal local oscillator of the R&S FSWP.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.loscillator.source.clone()
```

Subgroups**6.11.13.1.1 External****class ExternalCls**

External commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.loscillator.source.external.clone()
```

Subgroups**6.11.13.1.1.1 Level****SCPI Commands**

```
INPut<InputIx>:LOScillator:SOURce:EXTernal:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → LowHigh

```
# SCPI: INPut<ip>:LOScillator:SOURce:EXTernal:LEVel
value: enums.LowHigh = driver.inputPy.loscillator.source.external.level.
↪get(inputIx = repcap.InputIx.Default)
```

This command selects the level of an external LO signal that is fed into the R&S FSWP.

INTRO_CMD_HELP: Prerequisites for this command

- Select additive noise or pulsed additive noise measurement (CONFigure:PNOise:MEASurement) .
- Select an external local oscillator (method RsFswp.InputPy.Loscillator.Source.set) .

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

level: HIGH LO signal with high level characteristics. LOW LO signal with low level characteristics.

set(level: LowHigh, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:LOSCillator:SOURce:EXternal:LEVel
driver.inputPy.losillator.source.external.level.set(level = enums.LowHigh.HIGH,
↪ inputIx = repcap.InputIx.Default)
```

This command selects the level of an external LO signal that is fed into the R&S FSWP.

INTRO_CMD_HELP: Prerequisites for this command

- Select additive noise or pulsed additive noise measurement (CONFigure:PNOise:MEASurement).
- Select an external local oscillator (method RsFswp.InputPy.Loscillator.Source.set) .

param level

HIGH LO signal with high level characteristics. LOW LO signal with low level characteristics.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.14 Sanalyzer

class SanalyzerCls

Sanalyzer commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.sanalyzer.clone()
```

Subgroups

6.11.14.1 Attenuation

SCPI Commands

```
INPut<InputIx>:SANAlyzer:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:SANalyzer:ATTenuation
value: float = driver.inputPy.sanalyzer.attenuation.get(inputIx = repcap.
↪InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

attenuation: No help available

set(attenuation: float, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:SANalyzer:ATTenuation
driver.inputPy.sanalyzer.attenuation.set(attenuation = 1.0, inputIx = repcap.
↪InputIx.Default)
```

No command help available

param attenuation

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.sanalyzer.attenuation.clone()
```

Subgroups

6.11.14.1.1 Auto

SCPI Commands

```
INPut<InputIx>:SANalyzer:ATTenuation:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:SANalyzer:ATTenuation:AUTO
driver.inputPy.sanalyzer.attenuation.auto.set(state = False, inputIx = repcap.
↪InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.15 Select

SCPI Commands

```
INPut<InputIx>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*source*: InputSource, *inputIx*=InputIx.Default) → None

```
# SCPI: INPut<ip>:SElect
driver.inputPy.select.set(source = enums.InputSource.ABBand, inputIx = repcap.
↪InputIx.Default)
```

This command selects the signal source for measurements, i.e. it defines which connector is used to input data to the R&S FSWP.

param source

RF Radio Frequency (‘RF INPUT’ connector) FIQ I/Q data file (selected by method RsFswp.InputPy.File.Path.set) For details, see ‘Basics on input from I/Q data files’.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.16 Terminator

SCPI Commands

```
INPut<InputIx>:TERMinator
```

class TerminatorCls

Terminator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx*=InputIx.Default) → bool

```
# SCPI: INPut<ip>:TERMinator
value: bool = driver.inputPy.terminator.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:TERMinator
driver.inputPy.terminator.set(state = False, inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.11.17 Uport

class UportCls

Uport commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.inputPy.uport.clone()
```

Subgroups

6.11.17.1 State

SCPI Commands

```
INPut<InputIx>:UPORt:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: INPut<ip>:UPORt:STATe
value: bool = driver.inputPy.uport.state.get(inputIx = repcap.InputIx.Default)
```

This command toggles the control lines of the user ports for the AUX PORT connector. This 9-pole SUB-D male connector is located on the rear panel of the R&S FSWP.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: ON | 1 User port is switched to INPut OFF | 0 User port is switched to OUTPut

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: INPut<ip>:UPORt:STATe
driver.inputPy.uport.state.set(state = False, inputIx = repcap.InputIx.Default)
```

This command toggles the control lines of the user ports for the AUX PORT connector. This 9-pole SUB-D male connector is located on the rear panel of the R&S FSWP.

param state

ON | 1 User port is switched to INPut OFF | 0 User port is switched to OUTPut

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.11.17.2 Value

SCPI Commands

```
INPut<InputIx>:UPORt:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → float

```
# SCPI: INPut<ip>:UPORt[:VALue]
value: float = driver.inputPy.uport.value.get(inputIx = repcap.InputIx.Default)
```

This command queries the control lines of the user ports. For details see method RsFswp.Output.Uport.Value.set.

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

level: bit values in hexadecimal format TTL type voltage levels (max. 5V) Range: #B00000000 to #B00111111

6.12 Instrument

SCPI Commands

```
INSTRument:ABORt
INSTRument:DELeTe
```

class InstrumentCls

Instrument commands group definition. 32 total commands, 8 Subgroups, 2 group commands

abort() → None

```
# SCPI: INSTRUMENT:ABORt
driver.instrument.abort()
```

No command help available

abort_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: INSTRUMENT:ABORt
driver.instrument.abort_with_opc()
```

No command help available

Same as abort, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

delete(channel_name: str) → None

```
# SCPI: INSTRUMENT:DELeTe
driver.instrument.delete(channel_name = r1)
```

This command deletes a channel. If you delete the last channel, the default ‘Phase Noise’ channel is activated.

param channel_name

String containing the name of the channel you want to delete. A channel must exist to delete it.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.clone()
```

Subgroups

6.12.1 Couple

class CoupleCls

Couple commands group definition. 21 total commands, 17 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.couple.clone()
```

Subgroups

6.12.1.1 AbImpedance

SCPI Commands

```
INSTRument:COUPle:ABIMpedance
```

class AbImpedanceCls

AbImpedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:ABIMpedance
value: enums.Synchronization = driver.instrument.couple.abImpedance.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:ABIMpedance
driver.instrument.couple.abImpedance.set(state = enums.Synchronization.ALL)
```

No command help available

```
param state
    No help available
```

6.12.1.2 AcDc

SCPI Commands

```
INSTRument:COUPle:ACDC
```

class AcDcCls

AcDc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:ACDC
value: enums.Synchronization = driver.instrument.couple.acDc.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:ACDC
driver.instrument.couple.acDc.set(state = enums.Synchronization.ALL)
```

No command help available

param state

No help available

6.12.1.3 Atten

SCPI Commands

`INSTRument:COUPle:ATTen`

class AttenCls

Atten commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:ATTen
value: enums.Synchronization = driver.instrument.couple.atten.get()
```

No command help available

return

state: No help available

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:ATTen
driver.instrument.couple.atten.set(state = enums.Synchronization.ALL)
```

No command help available

param state

No help available

6.12.1.4 Aunit

SCPI Commands

`INSTRument:COUPle:AUNit`

class AunitCls

Aunit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:AUNit
value: enums.Synchronization = driver.instrument.couple.aunit.get()
```

No command help available

return

state: No help available

set(*state: Synchronization*) → None

```
# SCPI: INSTRument:COUPle:AUNit
driver.instrument.couple.aunit.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.1.5 Bandwidth

SCPI Commands

```
INSTRument:COUPle:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:BWIDth
value: enums.Synchronization = driver.instrument.couple.bandwidth.get()
```

No command help available

return
state: No help available

set(*state: Synchronization*) → None

```
# SCPI: INSTRument:COUPle:BWIDth
driver.instrument.couple.bandwidth.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.1.6 Center

SCPI Commands

```
INSTRument:COUPle:CENTer
```

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:CENTer
value: enums.Synchronization = driver.instrument.couple.center.get()
```

No command help available

return

state: No help available

set(state: *Synchronization*) → None

```
# SCPI: INSTRUMENT:COUPLE:CENter
driver.instrument.couple.center.set(state = enums.Synchronization.ALL)
```

No command help available

param state

No help available

6.12.1.7 Demod

SCPI Commands

```
INSTRUMENT:COUPLE:DEMod
```

class DemodCls

Demod commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → *Synchronization*

```
# SCPI: INSTRUMENT:COUPLE:DEMod
value: enums.Synchronization = driver.instrument.couple.demod.get()
```

No command help available

return

state: No help available

set(state: *Synchronization*) → None

```
# SCPI: INSTRUMENT:COUPLE:DEMod
driver.instrument.couple.demod.set(state = enums.Synchronization.ALL)
```

No command help available

param state

No help available

6.12.1.8 Gain

SCPI Commands

```
INSTRUMENT:COUPLE:GAIN
```

class GainCls

Gain commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → *Synchronization*

```
# SCPI: INSTRUMENT:COUPLE:GAIN
value: enums.Synchronization = driver.instrument.couple.gain.get()
```

No command help available

return

state: No help available

set(state: *Synchronization*) → None

```
# SCPI: INSTRument:COUPle:GAIN
driver.instrument.couple.gain.set(state = enums.Synchronization.ALL)
```

No command help available

param state

No help available

6.12.1.9 Generator

class GeneratorCls

Generator commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.couple.generator.clone()
```

Subgroups

6.12.1.9.1 Center

class CenterCls

Center commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.couple.generator.center.clone()
```

Subgroups

6.12.1.9.1.1 Offset

SCPI Commands

```
INSTRument:COUPle:GENerator:CENTer:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:CENTER:OFFSET
value: float = driver.instrument.couple.generator.center.offset.get()
```

No command help available

```
return
    frequency: No help available
```

set(frequency: float) → None

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:CENTER:OFFSET
driver.instrument.couple.generator.center.offset.set(frequency = 1.0)
```

No command help available

```
param frequency
    No help available
```

6.12.1.9.1.2 State

SCPI Commands

```
INSTRUMENT:COUPLE:GENERATOR:CENTER:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:CENTER[:STATE]
value: bool = driver.instrument.couple.generator.center.state.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: bool) → None

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:CENTER[:STATE]
driver.instrument.couple.generator.center.state.set(arg_0 = False)
```

No command help available

```
param arg_0
    No help available
```


6.12.1.9.2 RefLevel

class RefLevelCls

RefLevel commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.couple.generator.refLevel.clone()
```

Subgroups

6.12.1.9.2.1 Offset

SCPI Commands

```
INSTRUMENT:COUPLE:GENERATOR:RLEVEL:OFFSET
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:RLEVEL:OFFSET
value: float = driver.instrument.couple.generator.refLevel.offset.get()
```

No command help available

return

level: No help available

set(level: float) → None

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:RLEVEL:OFFSET
driver.instrument.couple.generator.refLevel.offset.set(level = 1.0)
```

No command help available

param level

No help available

6.12.1.9.2.2 State

SCPI Commands

```
INSTRUMENT:COUPLE:GENERATOR:RLEVEL:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:RLEVEL[:STATE]
value: bool = driver.instrument.couple.generator.refLevel.state.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: bool) → None

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:RLEVEL[:STATE]
driver.instrument.couple.generator.refLevel.state.set(arg_0 = False)
```

No command help available

```
param arg_0
    No help available
```

6.12.1.9.3 State

SCPI Commands

```
INSTRUMENT:COUPLE:GENERATOR:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:STATE
value: bool = driver.instrument.couple.generator.state.get()
```

No command help available

```
return
    arg_0: No help available
```

set(arg_0: bool) → None

```
# SCPI: INSTRUMENT:COUPLE:GENERATOR:STATE
driver.instrument.couple.generator.state.set(arg_0 = False)
```

No command help available

```
param arg_0
    No help available
```

6.12.1.10 Impedance

SCPI Commands

```
INSTRument:COUPle:IMPedance
```

class ImpedanceCls

Impedance commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:IMPedance
value: enums.Synchronization = driver.instrument.couple.impedance.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:IMPedance
driver.instrument.couple.impedance.set(state = enums.Synchronization.ALL)
```

No command help available

```
param state
    No help available
```

6.12.1.11 Limit

SCPI Commands

```
INSTRument:COUPle:LIMit
```

class LimitCls

Limit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:LIMit
value: enums.Synchronization = driver.instrument.couple.limit.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:LIMit
driver.instrument.couple.limit.set(state = enums.Synchronization.ALL)
```

No command help available

```
param state
    No help available
```

6.12.1.12 Llines

SCPI Commands

`INSTRument:COUPle:LLINes`

class LlinesCls

Llines commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:LLINes
value: enums.Synchronization = driver.instrument.couple.llines.get()
```

No command help available

return
state: No help available

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:LLINes
driver.instrument.couple.llines.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.1.13 Marker

SCPI Commands

`INSTRument:COUPle:MARKer`

class MarkerCls

Marker commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:MARKer
value: enums.Synchronization = driver.instrument.couple.marker.get()
```

No command help available

return
state: No help available

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:MARKer
driver.instrument.couple.marker.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.1.14 Presel

SCPI Commands

```
INSTRument:COUPle:PRESel
```

class PreselCls

Presel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:PRESel
value: enums.Synchronization = driver.instrument.couple.presel.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:PRESel
driver.instrument.couple.presel.set(state = enums.Synchronization.ALL)
```

No command help available

```
param state
    No help available
```

6.12.1.15 RefLevel

SCPI Commands

```
INSTRument:COUPle:RLEVel
```

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:RLEVel
value: enums.Synchronization = driver.instrument.couple.refLevel.get()
```

No command help available

```
return
    state: No help available
```

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:RLEVel
driver.instrument.couple.refLevel.set(state = enums.Synchronization.ALL)
```

No command help available

```
param state
    No help available
```

6.12.1.16 Span

SCPI Commands

`INSTRument:COUPle:SPAN`

class SpanCls

Span commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:SPAN
value: enums.Synchronization = driver.instrument.couple.span.get()
```

No command help available

return
state: No help available

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:SPAN
driver.instrument.couple.span.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.1.17 Vbw

SCPI Commands

`INSTRument:COUPle:VBW`

class VbwCls

Vbw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Synchronization

```
# SCPI: INSTRument:COUPle:VBW
value: enums.Synchronization = driver.instrument.couple.vbw.get()
```

No command help available

return
state: No help available

set(state: Synchronization) → None

```
# SCPI: INSTRument:COUPle:VBW
driver.instrument.couple.vbw.set(state = enums.Synchronization.ALL)
```

No command help available

param state
No help available

6.12.2 Create

class CreateCls

Create commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.instrument.create.clone()
```

Subgroups

6.12.2.1 Duplicate

SCPI Commands

```
INSTRUMENT:CREate:DUPLICATE
```

class DuplicateCls

Duplicate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(opc_timeout_ms: int = -1) → None

```
# SCPI: INSTRUMENT:CREate:DUPLICATE
driver.instrument.create.duplicate.set()
```

This command duplicates the currently selected channel, i.e creates a new channel of the same type and with the identical measurement settings. The name of the new channel is the same as the copied channel, extended by a consecutive number (e. g. 'IQAnalyzer' -> 'IQAnalyzer 2'). The channel to be duplicated must be selected first using the INST:SEL command. (See method RsFswp.Instrument.Select.set) . This command is not available if the MSRA primary channel is selected.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.12.2.2 New

SCPI Commands

```
INSTRUMENT:CREate:NEW
```

class NewCls

New commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(channel_type: ChannelType, channel_name: str) → None

```
# SCPI: INSTRUMENT:CREate[:NEW]
driver.instrument.create.new.set(channel_type = enums.ChannelType.IqAnalyzer=IQ,
↪ channel_name = '1')
```

This command adds a measurement channel. You can configure up to 10 measurement channels at the same time (depending on available memory) .

INTRO_CMD_HELP: See also

- method RsFswp.Instrument.Select.set
- method RsFswp.Instrument.delete

param channel_type

(enum or string) Channel type of the new channel. For a list of available channel types, see method **RsFswp.Instrument.ListPy.get_**.

param channel_name

String containing the name of the channel. Note that you cannot assign an existing channel name to a new channel. If you do, an error occurs.

6.12.2.3 Replace

SCPI Commands

INSTRument:CREate:REPLace

class ReplaceCls

Replace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(current_channel_name: str, channel_type: ChannelType, new_channel_name: str) → None

```
# SCPI: INSTRument:CREate:REPLace
driver.instrument.create.replace.set(current_channel_name = '1', channel_type =  
enums.ChannelType.IqAnalyzer=IQ, new_channel_name = '1')
```

This command replaces a channel with another one.

param current_channel_name

No help available

param channel_type

Channel type of the new channel. For a list of available channel types, see method **RsFswp.Instrument.ListPy.get_**.

param new_channel_name

No help available

6.12.3 ListPy

SCPI Commands

INSTRument:LIST

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: INSTRUMENT:LIST
value: List[str] = driver.instrument.listPy.get()
```

This command queries all active channels. The query is useful to obtain the names of the existing channels, which are required to replace or delete the channels.

return
result: No help available

6.12.4 Mode

SCPI Commands

INSTRUMENT:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → InstrumentMode

```
# SCPI: INSTRUMENT:MODE
value: enums.InstrumentMode = driver.instrument.mode.get()
```

No command help available

return
op_mode: No help available

set(op_mode: InstrumentMode) → None

```
# SCPI: INSTRUMENT:MODE
driver.instrument.mode.set(op_mode = enums.InstrumentMode.MSRanalyzer)
```

No command help available

param op_mode
No help available

6.12.5 Nselect

SCPI Commands

INSTRUMENT:NSELECT

class NselectCls

Nselect commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: INSTRUMENT:NSELECT
value: int = driver.instrument.nselect.get()
```

No command help available

```
    return
    option_number: No help available
```

set(*option_number: int*) → None

```
# SCPI: INSTRument:NSElect
driver.instrument.nselect.set(option_number = 1)
```

No command help available

```
    param option_number
    No help available
```

6.12.6 Rename

SCPI Commands

INSTRument:REName

class RenameCls

Rename commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*channel_name_1: str, channel_name_2: str*) → None

```
# SCPI: INSTRument:REName
driver.instrument.rename.set(channel_name_1 = '1', channel_name_2 = '1')
```

This command renames a channel.

```
    param channel_name_1
    String containing the name of the channel you want to rename.
```

```
    param channel_name_2
    String containing the new channel name. Note that you cannot assign an existing chan-
    nel name to a new channel. If you do, an error occurs. Channel names can have a
    maximum of 31 characters, and must be compatible with the Windows conventions for
    file names. In particular, they must not contain special characters such as '.', '*', '?'.
```

6.12.7 Select

SCPI Commands

INSTRument:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*channel_type: ChannelType*) → None

```
# SCPI: INSTRument[:SElect]
driver.instrument.select.set(channel_type = enums.ChannelType.IqAnalyzer=IQ)
```

This command activates a new channel with the defined channel type, or selects an existing channel with the specified name. Also see

INTRO_CMD_HELP: See also

- method `RsFswp.Instrument.Create.New.set`
- ‘Programming example: performing a sequence of measurements’

param channel_type

(enum or string) Channel type of the new channel. For a list of available channel types see method `RsFswp.Instrument.ListPy.get_`.

6.12.8 SelectName

SCPI Commands

```
INSTRument:SElect
```

class SelectNameCls

SelectName commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*channel_name: str*) → None

```
# SCPI: INSTRument[:SElect]
driver.instrument.selectName.set(channel_name = '1')
```

This command activates a new channel with the defined channel type, or selects an existing channel with the specified name. Also see

INTRO_CMD_HELP: See also

- method `RsFswp.Instrument.Create.New.set`
- ‘Programming example: performing a sequence of measurements’

param channel_name

String containing the name of the channel.

6.13 Layout

class LayoutCls

Layout commands group definition. 7 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.clone()
```

Subgroups

6.13.1 Add

class AddCls

Add commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.add.clone()
```

Subgroups

6.13.1.1 Window

SCPI Commands

```
LAYout:ADD:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str, direction: WindowDirection, window_type: WindowTypeBase) → str

```
# SCPI: LAYout:ADD[:WINDow]
value: str = driver.layout.add.window.get(window_name = '1', direction = enums.
↳ WindowDirection.ABOVe, window_type = enums.WindowTypeBase.Diagram=DIAGram)
```

This command adds a window to the display in the active channel. This command is always used as a query so that you immediately obtain the name of the new window as a result. To replace an existing window, use the method `RsFswp.Layout.Replace.Window.set` command.

param window_name

String containing the name of the existing window the new window is inserted next to. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method `RsFswp.Layout.Catalog.Window.get_query`.

param direction

LEFT | RIGHT | ABOVe | BELow Direction the new window is added relative to the existing window.

param window_type

(enum or string) text value Type of result display (evaluation method) you want to add. See the table below for available parameter values.

return

new_window_name: When adding a new window, the command returns its name (by default the same as its number) as a result.

6.13.2 Catalog

class CatalogCls

Catalog commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.catalog.clone()
```

Subgroups

6.13.2.1 Window

SCPI Commands

```
LAYout:CATalog:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: LAYout:CATalog[:WINDow]
value: List[str] = driver.layout.catalog.window.get()
```

This command queries the name and index of all active windows in the active channel from top left to bottom right. The result is a comma-separated list of values for each window, with the syntax: <Window-Name_1>,<WindowIndex_1>.. <WindowName_n>,<WindowIndex_n>

return

result: No help available

6.13.3 Direction

SCPI Commands

```
LAYout:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → XyDirection

```
# SCPI: LAYout:DIRection
value: enums.XyDirection = driver.layout.direction.get()
```

No command help available

return
direction: No help available

6.13.4 Identify

class IdentifyCls

Identify commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.identify.clone()
```

Subgroups

6.13.4.1 Window

SCPI Commands

```
LAYout:IDENtify:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window_name: str) → int

```
# SCPI: LAYout:IDENtify[:WINDow]
value: int = driver.layout.identify.window.get(window_name = '1')
```

This command queries the index of a particular display window in the active channel. Note: to query the name of a particular window, use the LAYout:WINDow<n>:IDENtify? query.

param window_name
String containing the name of a window.

return
window_index: Index number of the window.

6.13.5 Remove

class RemoveCls

Remove commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.remove.clone()
```

Subgroups

6.13.5.1 Window

SCPI Commands

```
LAYout:REMove:WINDow
```

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str) → None

```
# SCPI: LAYout:REMove[:WINDow]
driver.layout.remove.window.set(window_name = '1')
```

This command removes a window from the display in the active channel.

param window_name

String containing the name of the window. In the default state, the name of the window is its index.

6.13.6 Replace

class ReplaceCls

Replace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.layout.replace.clone()
```

Subgroups

6.13.6.1 Window

SCPI Commands

LAYout:REPLace:WINDow

class WindowCls

Window commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(window_name: str, window_type: WindowTypeBase) → None

```
# SCPI: LAYout:REPLace[:WINDow]
driver.layout.replace.window.set(window_name = '1', window_type = enums.
↳ WindowTypeBase.Diagram=DIAGram)
```

This command replaces the window type (for example from ‘Diagram’ to ‘Result Summary’) of an already existing window in the active channel while keeping its position, index and window name. To add a new window, use the method `RsFswp.Layout.Add.Window.get_` command.

param window_name

String containing the name of the existing window. By default, the name of a window is the same as its index. To determine the name and index of all active windows in the active channel, use the method `RsFswp.Layout.Catalog.Window.get_` query.

param window_type

(enum or string) Type of result display you want to use in the existing window. See method `RsFswp.Layout.Add.Window.get_` for a list of available window types.

6.13.7 Splitter

SCPI Commands

LAYout:SPLitter

class SplitterCls

Splitter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(index_1: int, index_2: int, position: int) → None

```
# SCPI: LAYout:SPLitter
driver.layout.splitter.set(index_1 = 1, index_2 = 1, position = 1)
```

This command changes the position of a splitter and thus controls the size of the windows on each side of the splitter. Compared to the method `RsFswp.Applications.K30_NoiseFigure.Display.Window.Size.set` command, the method `RsFswp.Applications.K30_NoiseFigure.Layout.Splitter.set` changes the size of all windows to either side of the splitter permanently, it does not just maximize a single window temporarily. Note that windows must have a certain minimum size. If the position you define conflicts with the minimum size of any of the affected windows, the command does not work, but does not return an error.

param index_1

The index of one window the splitter controls.

param index_2

The index of a window on the other side of the splitter.

param position

New vertical or horizontal position of the splitter as a fraction of the screen area (without channel and status bar and softkey menu). The point of origin (x = 0, y = 0) is in the lower left corner of the screen. The end point (x = 100, y = 100) is in the upper right corner of the screen. (See Figure 'SmartGrid coordinates for remote control of the splitters'.) The direction in which the splitter is moved depends on the screen layout. If the windows are positioned horizontally, the splitter also moves horizontally. If the windows are positioned vertically, the splitter also moves vertically. Range: 0 to 100

6.14 MassMemory

SCPI Commands

```
MMEMory:CLear:ALL
MMEMory:COPY
MMEMory:MOVE
```

class MassMemoryCls

MassMemory commands group definition. 69 total commands, 14 Subgroups, 3 group commands

clear_all(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:CLear:ALL
driver.massMemory.clear_all()
```

This command deletes all instrument configuration files in the current directory. You can select the directory with method RsFswp.MassMemory.CurrentDirectory.set.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

copy(source_file: str, target_file: str) → None

```
# SCPI: MMEMory:COPY
driver.massMemory.copy(source_file = '1', target_file = '1')
```

This command copies one or more files to another directory.

param source_file

No help available

param target_file

No help available

move(source_file: str, target_file: str) → None

```
# SCPI: MMEMory:MOVE
driver.massMemory.move(source_file = '1', target_file = '1')
```

This command moves a file to another directory. The command also renames the file if you define a new name in the target directory. If you do not include a path for <NewFileName>, the command just renames the file.

param source_file
No help available

param target_file
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.clone()
```

Subgroups

6.14.1 Catalog

SCPI Commands

```
MMEMory:CATalog
```

class CatalogCls

Catalog commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:CATalog
value: str = driver.massMemory.catalog.get()
```

This command returns the contents of a particular directory.

return

filename: String containing the path and directory If you leave out the path, the command returns the contents of the directory selected with method RsFswp.MassMemory.CurrentDirectory.set. The path may be relative or absolute. Using wildcards ('*') is possible to query a certain type of files only. If you use a specific file as a parameter, the command returns the name of the file if the file is found in the specified directory, or an error if the file is not found ('-256,'File name not found').

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.catalog.clone()
```

Subgroups

6.14.1.1 Long

SCPI Commands

```
MMEMory:CATalog:LONG
```

class LongCls

Long commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:CATalog:LONG
value: str = driver.massMemory.catalog.long.get()
```

This command returns the contents of a particular directory with additional information about the files.

return

directory: String containing the path and directory. If you leave out the path, the command returns the contents of the directory selected with method RsFswp.MassMemory.CurrentDirectory.set. The path may be relative or absolute. Using wildcards ('*') is possible to query a certain type of files only.

set(directory: str) → None

```
# SCPI: MMEMory:CATalog:LONG
driver.massMemory.catalog.long.set(directory = '1')
```

This command returns the contents of a particular directory with additional information about the files.

param directory

String containing the path and directory. If you leave out the path, the command returns the contents of the directory selected with method RsFswp.MassMemory.CurrentDirectory.set. The path may be relative or absolute. Using wildcards ('*') is possible to query a certain type of files only.

6.14.2 Clear

class ClearCls

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.clear.clone()
```

Subgroups

6.14.2.1 State

SCPI Commands

```
MMEMory:CLear:STATe 1,
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: MMEMory:CLear:STATe
driver.massMemory.clear.state.set(filename = '1')
```

This command deletes an instrument configuration file.

param filename

String containing the path and name of the file to delete. The string may or may not contain the file's extension.

6.14.3 Comment

SCPI Commands

```
MMEMory:COMMeNt
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:COMMeNt
value: str = driver.massMemory.comment.get()
```

This command defines a comment for the stored settings.

return

comment: String containing the comment.

set(comment: str) → None

```
# SCPI: MMEMory:COMMeNt
driver.massMemory.comment.set(comment = '1')
```

This command defines a comment for the stored settings.

param comment

String containing the comment.

6.14.4 CurrentDirectory

SCPI Commands

```
MMEMory:CDIRectory
```

class CurrentDirectoryCls

CurrentDirectory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*directory: str*) → None

```
# SCPI: MMEMory:CDIRectory
driver.massMemory.currentDirectory.set(directory = '1')
```

This command changes the current directory.

param directory

String containing the path to another directory. The path may be relative or absolute.

6.14.5 Delete

class DeleteCls

Delete commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.delete.clone()
```

Subgroups

6.14.5.1 Immediate

SCPI Commands

```
MMEMory:DELeTe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*filename: str*) → None

```
# SCPI: MMEMory:DELeTe:IMMediate
driver.massMemory.delete.immediate.set(filename = '1')
```

This command deletes a file.

param filename

String containing the path and file name of the file to delete. The path may be relative or absolute.

6.14.6 DeleteDirectory

SCPI Commands

MMEMory:RDIRECTory

class DeleteDirectoryCls

DeleteDirectory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:RDIRECTory
value: str = driver.massMemory.deleteDirectory.get()
```

This command deletes the indicated directory.

return

arg_0: String containing the path of the directory to delete. Note that the directory you want to remove must be empty.

set(arg_0: str) → None

```
# SCPI: MMEMory:RDIRECTory
driver.massMemory.deleteDirectory.set(arg_0 = '1')
```

This command deletes the indicated directory.

param arg_0

String containing the path of the directory to delete. Note that the directory you want to remove must be empty.

6.14.7 Load<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.massMemory.load.repcap_window_get()
driver.massMemory.load.repcap_window_set(repcap.Window.Nr1)
```

class LoadCls

Load commands group definition. 8 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.load.clone()
```

Subgroups

6.14.7.1 Auto

SCPI Commands

```
MMEMory:LOAD:AUTO 1,
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: MMEMory:LOAD:AUTO
driver.massMemory.load.auto.set(filename = '1')
```

This command restores an instrument configuration and defines that configuration as the default state. The default state is restored after a preset (*RST) or after you turn on the R&S FSWP.

param filename

‘Factory’ Restores the factory settings as the default state. ‘file_name String containing the path and name of the configuration file. Note that only instrument settings files can be selected for the startup recall function; channel files cause an error.

6.14.7.2 Iq

class IqCls

Iq commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.load.iq.clone()
```

Subgroups

6.14.7.2.1 State

SCPI Commands

```
MMEMory:LOAD:IQ:STATe 1,
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: MMEMory:LOAD:IQ:STATe
driver.massMemory.load.iq.state.set(filename = '1')
```

This command restores I/Q data from a file. The file extension is *.iq.tar.

param filename

string String containing the path and name of the source file.

6.14.7.2.2 Stream**SCPI Commands**

MMEMory:LOAD:IQ:STReam

class StreamCls

Stream commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEMory:LOAD:IQ:STReam
value: str = driver.massMemory.load.iq.stream.get()
```

No command help available

return

channel: No help available

set(channel: str) → None

```
# SCPI: MMEMory:LOAD:IQ:STReam
driver.massMemory.load.iq.stream.set(channel = '1')
```

No command help available

param channel

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.load.iq.stream.clone()
```

Subgroups**6.14.7.2.2.1 Auto****SCPI Commands**

MMEMory:LOAD:IQ:STReam:AUTO

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None


```
# SCPI: MMEemory:LOAD:IQ:STReam:AUTO
driver.massMemory.load.iq.stream.auto.set(state = False)
```

No command help available

param state

No help available

6.14.7.2.2 ListPy

SCPI Commands

```
MMEemory:LOAD:IQ:STReam:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: MMEemory:LOAD:IQ:STReam:LIST
value: List[str] = driver.massMemory.load.iq.stream.listPy.get()
```

No command help available

return

result: No help available

6.14.7.3 State

SCPI Commands

```
MMEemory:LOAD:STATe 1,
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: MMEemory:LOAD:STATe
driver.massMemory.load.state.set(filename = '1')
```

This command restores and activates the instrument configuration stored in a *.dfl file. Note that files with other formats cannot be loaded with this command. The contents that are reloaded from the file are defined by the last selection made either in the ‘Save/Recall’ dialogs (manual operation) or through the MMEemory:SElect[:ITEM] commands (remote operation; the settings are identical in both cases) . By default, the selection is limited to the user settings (‘User Settings’ selection in the dialogs, HWSettings in SCPI) . The selection is not reset by [Preset] or *RST. As a consequence, the results of a SCPI script using the method RsFswp.MassMemory.Load.State.set command without a previous MMEemory:SElect[:ITEM] command may vary, depending on previous actions in the GUI or in previous scripts, even if the script starts with the *RST command. It is therefore recommended that you use the appropriate MMEemory:SElect[:ITEM] command before using method RsFswp.MassMemory.Load.State.set.

param filename

String containing the path and name of the file to load. The string may or may not include the file's extension.

6.14.7.4 Tfactor

SCPI Commands

MMEMory:LOAD<Window>:TFACTOR

class TfactorCls

Tfactor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: MMEMory:LOAD<n>:TFACTOR
value: str = driver.massMemory.load.tfactor.get(window = repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Load')

return

filename: No help available

set(filename: str, window=Window.Default) → None

```
# SCPI: MMEMory:LOAD<n>:TFACTOR
driver.massMemory.load.tfactor.set(filename = '1', window = repcap.Window.
↪Default)
```

No command help available

param filename

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Load')

6.14.7.5 TypePy

SCPI Commands

MMEMory:LOAD:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → LoadType

```
# SCPI: MMEMory:LOAD:TYPE
value: enums.LoadType = driver.massMemory.load.typePy.get()
```

No command help available

return

type_py: No help available

set(type_py: LoadType) → None

```
# SCPI: MMEemory:LOAD:TYPE
driver.massMemory.load.typePy.set(type_py = enums.LoadType.NEW)
```

No command help available

param type_py

No help available

6.14.8 MakeDirectory

SCPI Commands

MMEemory:MDIRectory

class MakeDirectoryCls

MakeDirectory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:MDIRectory
value: str = driver.massMemory.makeDirectory.get()
```

This command creates a new directory.

return

directory: String containing the path and new directory name The path may be relative or absolute.

set(directory: str) → None

```
# SCPI: MMEemory:MDIRectory
driver.massMemory.makeDirectory.set(directory = '1')
```

This command creates a new directory.

param directory

String containing the path and new directory name The path may be relative or absolute.

6.14.9 Msis

SCPI Commands

MMEemory:MSIS

class MsisCls

Msis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:MSIS
value: str = driver.massMemory.msis.get()
```

This command selects the default storage device used by all MMEemory commands.

return
drive: No help available

set(drive: str) → None

```
# SCPI: MMEemory:MSIS
driver.massMemory.msis.set(drive = '1')
```

This command selects the default storage device used by all MMEemory commands.

param drive
'A:' | 'C:' | ... | 'Z:' String containing the device drive name

6.14.10 Name

SCPI Commands

MMEemory:NAME

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:NAME
value: str = driver.massMemory.name.get()
```

This command has several purposes, depending on the context it is used in.

- It creates a new and empty file.
- It defines the file name for screenshots taken with method RsFswp.HardCopy.Immediate.set. Note that you have to route the printer output to a file.

return
filename: String containing the path and name of the target file.

set(filename: str) → None

```
# SCPI: MMEemory:NAME
driver.massMemory.name.set(filename = '1')
```

This command has several purposes, depending on the context it is used in.

- It creates a new and empty file.
- It defines the file name for screenshots taken with method RsFswp.HardCopy.Immediate.set. Note that you have to route the printer output to a file.

param filename

String containing the path and name of the target file.

6.14.11 Network

class NetworkCls

Network commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.network.clone()
```

Subgroups

6.14.11.1 Disconnect

SCPI Commands

```
MMEMory:NETWork:DISConnect
```

class DisconnectCls

Disconnect commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DisconnectStruct

Response structure. Fields:

- Drive: str: String containing the drive name.
- State: bool: 1 | 0 | ON | OFF Optional: determines whether disconnection is forced or not 1 | ON Disconnection is forced. 0 | OFF Disconnect only if not in use.

get() → DisconnectStruct

```
# SCPI: MMEMory:NETWork:DISConnect
value: DisconnectStruct = driver.massMemory.network.disconnect.get()
```

This command disconnects a network drive.

return

structure: for return value, see the help for DisconnectStruct structure arguments.

set(drive: str, state: Optional[bool] = None) → None

```
# SCPI: MMEMory:NETWork:DISConnect
driver.massMemory.network.disconnect.set(drive = '1', state = False)
```

This command disconnects a network drive.

param drive

String containing the drive name.

param state

1 | 0 | ON | OFF Optional: determines whether disconnection is forced or not 1 | ON
Disconnection is forced. 0 | OFF Disconnect only if not in use.

6.14.11.2 Map

SCPI Commands

MMEMory:NETWork:MAP

class MapCls

Map commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MapStruct

Response structure. Fields:

- File_Path: str: String containing the drive name or path of the directory you want to map.
- Ip: str: String containing the host name of the computer or the IP address and the share name of the drive. '/host name or IP address/share name'
- User_Name: str: String containing a user name in the network. The user name is optional.
- Password: str: String containing the password corresponding to the UserName. The password is optional.
- State: bool: ON | OFF | 1 | 0 ON | 1 Reconnects at logon with the same user name. OFF | 0 Does not reconnect at logon.

get() → MapStruct

```
# SCPI: MMEMory:NETWork:MAP
value: MapStruct = driver.massMemory.network.map.get()
```

This command maps a drive to a server or server directory of the network. Note that you have to allow sharing for a server or folder in Microsoft networks first.

return

structure: for return value, see the help for MapStruct structure arguments.

set(file_path: str, ip: str, user_name: Optional[str] = None, password: Optional[str] = None, state: Optional[bool] = None) → None

```
# SCPI: MMEMory:NETWork:MAP
driver.massMemory.network.map.set(file_path = '1', ip = '1', user_name = '1',
↳ password = '1', state = False)
```

This command maps a drive to a server or server directory of the network. Note that you have to allow sharing for a server or folder in Microsoft networks first.

param file_path

String containing the drive name or path of the directory you want to map.

param ip

String containing the host name of the computer or the IP address and the share name of the drive. '/host name or IP address/share name'

param user_name

String containing a user name in the network. The user name is optional.

param password

String containing the password corresponding to the UserName. The password is optional.

param state

ON | OFF | 1 | 0 ON | 1 Reconnects at logon with the same user name. OFF | 0 Does not reconnect at logon.

6.14.11.3 UnusedDrives**SCPI Commands**

```
MMEMory:NETWork:UNUSeddrives
```

class UnusedDrivesCls

UnusedDrives commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: MMEMory:NETWork:UNUSeddrives
value: List[str] = driver.massMemory.network.unusedDrives.get()
```

This command returns a list of unused network drives.

return

drives: List of network drives in alphabetically descending order, e.g. 'W:,V:,U:,...'

6.14.11.4 UsedDrives**SCPI Commands**

```
MMEMory:NETWork:USEDdrives
```

class UsedDrivesCls

UsedDrives commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:NETWork:USEDdrives
value: bool = driver.massMemory.network.usedDrives.get()
```

This command returns a list of all network drives in use.

return

state: You do not have to use the parameter. If you do not include the parameter, the command returns a list of all drives in use. This is the same behavior as if you were using the parameter OFF. ON | 1 Returns a list of all drives in use including the folder information. OFF | 0 Returns a list of all drives in use.

set(*state: Optional[bool] = None*) → None

```
# SCPI: MMEemory:NETWork:USEDdrives
driver.massMemory.network.usedDrives.set(state = False)
```

This command returns a list of all network drives in use.

param state

You do not have to use the parameter. If you do not include the parameter, the command returns a list of all drives in use. This is the same behavior as if you were using the parameter OFF. ON | 1 Returns a list of all drives in use including the folder information. OFF | 0 Returns a list of all drives in use.

6.14.12 Raw

SCPI Commands

```
MMEemory:RAW
```

class RawCls

Raw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: MMEemory:RAW
value: str = driver.massMemory.raw.get()
```

No command help available

return

path: No help available

set(*path: str*) → None

```
# SCPI: MMEemory:RAW
driver.massMemory.raw.set(path = '1')
```

No command help available

param path

No help available

6.14.13 Select

class SelectCls

Select commands group definition. 28 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.clone()
```

Subgroups

6.14.13.1 Channel

class ChannelCls

Channel commands group definition. 10 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.channel.clone()
```

Subgroups

6.14.13.1.1 Item

class ItemCls

Item commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.channel.item.clone()
```

Subgroups

6.14.13.1.1.1 All

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:ALL
driver.massMemory.select.channel.item.all.set()
```

This command includes all items when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded. The items are:

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:ALL
driver.massMemory.select.channel.item.all.set_with_opc()
```

This command includes all items when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded. The items are:

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.1.1.2 Default

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:DEFault
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:DEFault
driver.massMemory.select.channel.item.default.set()
```

This command selects the current settings as the only item to store to and load from a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:DEFault
driver.massMemory.select.channel.item.default.set_with_opc()
```

This command selects the current settings as the only item to store to and load from a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.1.1.3 HwSettings

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:HwSettings
```

class HwSettingsCls

HwSettings commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:HwSettings
value: bool = driver.massMemory.select.channel.item.hwSettings.get()
```

This command includes or excludes measurement (hardware) settings when storing or loading a configuration file. Measurement settings include:

- general channel configuration
- measurement hardware configuration including markers
- limit lines Note that a configuration may include no more than 8 limit lines. This number includes active limit lines as well as inactive limit lines that were used last. Therefore the combination of inactivate limit lines depends on the sequence of use with method RsFswp.MassMemory.Load.State.set.
- color settings
- configuration for the hardcopy output

Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:HwSettings
driver.massMemory.select.channel.item.hwSettings.set(state = False)
```

This command includes or excludes measurement (hardware) settings when storing or loading a configuration file. Measurement settings include:

- general channel configuration
- measurement hardware configuration including markers

- limit lines Note that a configuration may include no more than 8 limit lines. This number includes active limit lines as well as inactive limit lines that were used last. Therefore the combination of inactive limit lines depends on the sequence of use with method RsFswp.MassMemory.Load.State.set.
- color settings
- configuration for the hardcopy output

Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state
ON | OFF | 0 | 1

6.14.13.1.1.4 Lines

class LinesCls

Lines commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.channel.item.lines.clone()
```

Subgroups

6.14.13.1.1.5 All

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:LINes:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:LINes:ALL
driver.massMemory.select.channel.item.lines.all.set(state = False)
```

This command includes or excludes all limit lines (active and inactive) when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state
ON | OFF | 1 | 0

6.14.13.1.1.6 NonePy

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:NONE
```

class NonePyCls

NonePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:NONE
driver.massMemory.select.channel.item.nonePy.set()
```

This command does not include any of the following items when storing or loading a configuration file.

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:NONE
driver.massMemory.select.channel.item.nonePy.set_with_opc()
```

This command does not include any of the following items when storing or loading a configuration file.

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.1.1.7 ScData

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:ScData
```

class ScDataCls

ScData commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEemory:SElect:CHANnel[:ITEM]:SCData
value: bool = driver.massMemory.select.channel.item.scData.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEemory:SElect:CHANnel[:ITEM]:SCData
driver.massMemory.select.channel.item.scData.set(state = False)
```

No command help available

param state
No help available

6.14.13.1.1.8 Spectrogram

SCPI Commands

```
MMEemory:SElect:CHANnel:ITEM:SPECTrogram
```

class SpectrogramCls

Spectrogram commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEemory:SElect:CHANnel[:ITEM]:SPECTrogram
value: bool = driver.massMemory.select.channel.item.spectrogram.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEemory:SElect:CHANnel[:ITEM]:SPECTrogram
driver.massMemory.select.channel.item.spectrogram.set(state = False)
```

No command help available

param state
No help available

6.14.13.1.1.9 Trace

class TraceCls

Trace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.channel.item.trace.clone()
```

Subgroups

6.14.13.1.1.10 Active

SCPI Commands

```
MMEMory:SElect:CHANnel[:ITEM]:TRACe:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:TRACe[:ACTive]
value: bool = driver.massMemory.select.channel.item.trace.active.get()
```

This command includes or excludes trace data when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:TRACe[:ACTive]
driver.massMemory.select.channel.item.trace.active.set(state = False)
```

This command includes or excludes trace data when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state

ON | OFF | 1 | 0

6.14.13.1.11 Transducer

class TransducerCls

Transducer commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.channel.item.transducer.clone()
```

Subgroups

6.14.13.1.12 All

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:TRANsducer:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:TRANsducer:ALL
driver.massMemory.select.channel.item.transducer.all.set(state = False)
```

This command includes or excludes transducer factors when storing or loading a configuration file. The command is available in the optional Spectrum application. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state
ON | OFF | 1 | 0

6.14.13.1.13 Weighting

SCPI Commands

```
MMEMory:SElect:CHANnel:ITEM:WEIGHting
```

class WeightingCls

Weighting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect:CHANnel[:ITEM]:WEIGHting
value: bool = driver.massMemory.select.channel.item.weighting.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEemory:SElect:CHANnel[:ITEM]:WEIGHting
driver.massMemory.select.channel.item.weighting.set(state = False)
```

No command help available

param state

No help available

6.14.13.2 Item

class ItemCls

Item commands group definition. 18 total commands, 16 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.item.clone()
```

Subgroups

6.14.13.2.1 All

SCPI Commands

MMEemory:SElect:ITEM:ALL

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEemory:SElect[:ITEM]:ALL
driver.massMemory.select.item.all.set()
```

This command includes all items when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEemory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEemory:SElect:CHANnel[:ITEM]...) are stored or loaded. The items are:

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEemory:SElect[:ITEM]:SGRam
- Trace data: MMEemory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:SElect[:ITEM]:ALL
driver.massMemory.select.item.all.set_with_opc()
```

This command includes all items when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded. The items are:

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTivE]

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.2.2 Cdata

SCPI Commands

```
MMEMory:SElect:ITEM:CDATa
```

class CdataCls

Cdata commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:CDATa
value: bool = driver.massMemory.select.item.cdata.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:CDATa
driver.massMemory.select.item.cdata.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.3 Csetup

SCPI Commands

```
MMEMory:SElect:ITEM:CSETup
```

class CsetupCls

Csetup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:CSETup
value: bool = driver.massMemory.select.item.csetup.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:CSETup
driver.massMemory.select.item.csetup.set(state = False)
```

No command help available

param state
No help available

6.14.13.2.4 Default

SCPI Commands

```
MMEMory:SElect:ITEM:DEFAult
```

class DefaultCls

Default commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEMory:SElect[:ITEM]:DEFAult
driver.massMemory.select.item.default.set()
```

This command selects the current settings as the only item to store to and load from a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:SElect[:ITEM]:DEFAult
driver.massMemory.select.item.default.set_with_opc()
```

This command selects the current settings as the only item to store to and load from a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.2.5 Final

SCPI Commands

MMEMory:SElect:ITEM:FINAl

class FinalCls

Final commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:FINAl
value: bool = driver.massMemory.select.item.final.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:FINAl
driver.massMemory.select.item.final.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.6 HardCopy

SCPI Commands

MMEMory:SElect:ITEM:HCOPy

class HardCopyCls

HardCopy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:HCOPy
value: bool = driver.massMemory.select.item.hardCopy.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:HCOPY
driver.massMemory.select.item.hardCopy.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.7 HwSettings

SCPI Commands

```
MMEMory:SElect:ITEM:HwSettings
```

class HwSettingsCls

HwSettings commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:HwSettings
value: bool = driver.massMemory.select.item.hwSettings.get()
```

This command includes or excludes measurement (hardware) settings when storing or loading a configuration file. Measurement settings include:

- general channel configuration
- measurement hardware configuration including markers
- limit lines Note that a configuration may include no more than 8 limit lines. This number includes active limit lines as well as inactive limit lines that were used last. Therefore the combination of inactivate limit lines depends on the sequence of use with method RsFswp.MassMemory.Load.State.set.
- color settings
- configuration for the hardcopy output

Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:HwSettings
driver.massMemory.select.item.hwSettings.set(state = False)
```

This command includes or excludes measurement (hardware) settings when storing or loading a configuration file. Measurement settings include:

- general channel configuration
- measurement hardware configuration including markers
- limit lines Note that a configuration may include no more than 8 limit lines. This number includes active limit lines as well as inactive limit lines that were used last. Therefore the combination of inactive limit lines depends on the sequence of use with method RsFswp.MassMemory.Load.State.set.
- color settings
- configuration for the hardcopy output

Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state
ON | OFF | 0 | 1

6.14.13.2.8 Lines

class LinesCls

Lines commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.item.lines.clone()
```

Subgroups

6.14.13.2.8.1 Active

SCPI Commands

```
MMEMory:SElect:ITEM:LINEs:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:LINEs[:ACTive]
value: bool = driver.massMemory.select.item.lines.active.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:LINEs[:ACTive]
driver.massMemory.select.item.lines.active.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.8.2 All

SCPI Commands

```
MMEMory:SElect:ITEM:LINes:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:LINes:ALL
driver.massMemory.select.item.lines.all.set(state = False)
```

This command includes or excludes all limit lines (active and inactive) when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state

ON | OFF | 1 | 0

6.14.13.2.9 Macros

SCPI Commands

```
MMEMory:SElect:ITEM:MACRos
```

class MacrosCls

Macros commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:MACRos
value: bool = driver.massMemory.select.item.macros.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:MACRos
driver.massMemory.select.item.macros.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.10 NonePy

SCPI Commands

`MMEMory:SElect:ITEM:NONE`

class NonePyCls

NonePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`set()` → None

```
# SCPI: MMEMory:SElect[:ITEM]:NONE
driver.massMemory.select.item.nonePy.set()
```

This command does not include any of the following items when storing or loading a configuration file.

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

`set_with_opc(opc_timeout_ms: int = -1)` → None

```
# SCPI: MMEMory:SElect[:ITEM]:NONE
driver.massMemory.select.item.nonePy.set_with_opc()
```

This command does not include any of the following items when storing or loading a configuration file.

- Hardware configuration: method RsFswp.MassMemory.Select.Item.HwSettings.set
- Limit lines: method RsFswp.MassMemory.Select.Item.Lines.All.set
- Spectrogram data: MMEMory:SElect[:ITEM]:SGRam
- Trace data: MMEMory:SElect[:ITEM]:TRACe<1...3>[:ACTive]

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.13.2.11 ScData

SCPI Commands

`MMEMory:SElect:ITEM:ScData`

class ScDataCls

ScData commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMemory:SElect[:ITEM]:SCData
value: bool = driver.massMemory.select.item.scData.get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: MMemory:SElect[:ITEM]:SCData
driver.massMemory.select.item.scData.set(state = False)
```

No command help available

```
param state
No help available
```

6.14.13.2.12 Spectrogram

SCPI Commands

```
MMemory:SElect:ITEM:SPECTrogram
```

class SpectrogramCls

Spectrogram commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMemory:SElect[:ITEM]:SPECTrogram
value: bool = driver.massMemory.select.item.spectrogram.get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: MMemory:SElect[:ITEM]:SPECTrogram
driver.massMemory.select.item.spectrogram.set(state = False)
```

No command help available

```
param state
No help available
```

6.14.13.2.13 Trace

class TraceCls

Trace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.item.trace.clone()
```

Subgroups

6.14.13.2.13.1 Active

SCPI Commands

```
MMEMory:SElect:ITEM:TRACe:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:TRACe[:ACTive]
value: bool = driver.massMemory.select.item.trace.active.get()
```

This command includes or excludes trace data when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:TRACe[:ACTive]
driver.massMemory.select.item.trace.active.set(state = False)
```

This command includes or excludes trace data when storing or loading a configuration file. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state

ON | OFF | 1 | 0

6.14.13.2.14 Transducer

class TransducerCls

Transducer commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.select.item.transducer.clone()
```

Subgroups

6.14.13.2.14.1 Active

SCPI Commands

```
MMEMory:SElect:ITEM:TRANsducer:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:TRANsducer[:ACTive]
value: bool = driver.massMemory.select.item.transducer.active.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:TRANsducer[:ACTive]
driver.massMemory.select.item.transducer.active.set(state = False)
```

No command help available

param state

No help available

6.14.13.2.14.2 All

SCPI Commands

```
MMEMory:SElect:ITEM:TRANsducer:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:TRANsducer:ALL
driver.massMemory.select.item.transducer.all.set(state = False)
```

This command includes or excludes transducer factors when storing or loading a configuration file. The command is available in the optional Spectrum application. Depending on the used command, either the items from the entire instrument (MMEMory:SElect[:ITEM]...) , or only those from the currently selected channel (MMEM:SElect:CHANnel[:ITEM]...) are stored or loaded.

param state
ON | OFF | 1 | 0

6.14.13.2.15 ViqData

SCPI Commands

```
MMEMory:SElect:ITEM:VIQData
```

class ViqDataCls

ViqData commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEMory:SElect[:ITEM]:VIQData
value: bool = driver.massMemory.select.item.viqData.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: MMEMory:SElect[:ITEM]:VIQData
driver.massMemory.select.item.viqData.set(state = False)
```

No command help available

param state
No help available

6.14.13.2.16 Weighting

SCPI Commands

```
MMEMory:SElect:ITEM:WEIGHting
```

class WeightingCls

Weighting commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: MMEemory:SElect[:ITEM]:WEIGHting
value: bool = driver.massMemory.select.item.weighting.get()
```

No command help available

```
return
state: No help available
```

set(state: bool) → None

```
# SCPI: MMEemory:SElect[:ITEM]:WEIGHting
driver.massMemory.select.item.weighting.set(state = False)
```

No command help available

```
param state
No help available
```

6.14.14 Store<Store>

RepCap Settings

```
# Range: Pos1 .. Pos32
rc = driver.massMemory.store.repcap_store_get()
driver.massMemory.store.repcap_store_set(repcap.Store.Pos1)
```

class StoreCls

Store commands group definition. 15 total commands, 10 Subgroups, 0 group commands Repeated Capability: Store, default value after init: Store.Pos1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.store.clone()
```

Subgroups

6.14.14.1 Iq

class IqCls

Iq commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.store.iq.clone()
```

Subgroups

6.14.14.1.1 Comment

SCPI Commands

```
MMEMory:STORe<Store>:IQ:COMMeNt
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → str

```
# SCPI: MMEMory:STORe<n>:IQ:COMMeNt
value: str = driver.massMemory.store.iq.comment.get(store = repcap.Store.
↳Default)
```

This command adds a comment to a file that contains I/Q data.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

comment: String containing the comment.

set(comment: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:IQ:COMMeNt
driver.massMemory.store.iq.comment.set(comment = '1', store = repcap.Store.
↳Default)
```

This command adds a comment to a file that contains I/Q data.

param comment

String containing the comment.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.1.2 FormatPy

SCPI Commands

```
MMEMory:STORe<Store>:IQ:FORMat
```

class FormatPyCls

FormatPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*store=Store.Default*) → IqDataFormat

```
# SCPI: MMEMory:STORe<n>:IQ:FORMat
value: enums.IqDataFormat = driver.massMemory.store.iq.formatPy.get(store = ↵
↵repcap.Store.Default)
```

No command help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

data_type: No help available

set(*data_type: IqDataFormat, store=Store.Default*) → None

```
# SCPI: MMEMory:STORe<n>:IQ:FORMat
driver.massMemory.store.iq.formatPy.set(data_type = enums.IqDataFormat.
↵FloatComplex=FL0at32,COMplex, store = repcap.Store.Default)
```

No command help available

param data_type

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.1.3 Range

SCPI Commands

```
MMEMory:STORe<Store>:IQ:RANGe
```

class RangeCls

Range commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*store=Store.Default*) → IqRangeType

```
# SCPI: MMEMory:STORe<n>:IQ:RANGe
value: enums.IqRangeType = driver.massMemory.store.iq.range.get(store = repcap.
↵Store.Default)
```

No command help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

range_type: No help available

set(range_type: *IqRangeType*, store=*Store.Default*) → None

```
# SCPI: MMEemory:STORE<n>:IQ:RANGe
driver.massMemory.store.iq.range.set(range_type = enums.IqRangeType.CAPture,
↵store = repcap.Store.Default)
```

No command help available

param range_type

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.1.4 State

SCPI Commands

```
MMEemory:STORE:IQ:STATE 1,
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: *str*) → None

```
# SCPI: MMEemory:STORE:IQ:STATE
driver.massMemory.store.iq.state.set(filename = '1')
```

This command writes the captured I/Q data to a file. The file extension is *.iq.tar. By default, the contents of the file are in 32-bit floating point format.

param filename

String containing the path and name of the target file.

6.14.14.2 ListPy

SCPI Commands

```
MMEemory:STORE<Store>:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=*Store.Default*) → *str*


```
# SCPI: MMEemory:STORe<n>:LIST
value: str = driver.massMemory.store.listPy.get(store = repcap.Store.Default)
```

This command exports the SEM and spurious emission list evaluation to a file. The file format is *.dat.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

filename: String containing the path and name of the target file.

6.14.14.3 Peak

SCPI Commands

```
MMEemory:STORe<Store>:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → str

```
# SCPI: MMEemory:STORe<n>:PEAK
value: str = driver.massMemory.store.peak.get(store = repcap.Store.Default)
```

No command help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

filename: No help available

set(filename: str, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:PEAK
driver.massMemory.store.peak.set(filename = '1', store = repcap.Store.Default)
```

No command help available

param filename

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.4 Spectrogram

SCPI Commands

```
MMEMory:STORe<Store>:SPECTrogram
```

class SpectrogramCls

Spectrogram commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → str

```
# SCPI: MMEMory:STORe<n>:SPECTrogram
value: str = driver.massMemory.store.spectrogram.get(store = repcap.Store.
↳Default)
```

No command help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

filename: No help available

set(filename: str, store=Store.Default) → None

```
# SCPI: MMEMory:STORe<n>:SPECTrogram
driver.massMemory.store.spectrogram.set(filename = '1', store = repcap.Store.
↳Default)
```

No command help available

param filename

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.5 Spurious

SCPI Commands

```
MMEMory:STORe<Store>:SPURious
```

class SpuriousCls

Spurious commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(store=Store.Default) → str

```
# SCPI: MMEMory:STORe<n>:SPURious
value: str = driver.massMemory.store.spurious.get(store = repcap.Store.Default)
```

This command exports the spur information to a file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a

‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device. For details, see ‘Protecting data using the secure user mode’.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

return

filename: String containing the path and name of the target file.

set(filename: str, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:SPURious
driver.massMemory.store.spurious.set(filename = '1', store = repcap.Store.
↳Default)
```

This command exports the spur information to a file. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a ‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device. For details, see ‘Protecting data using the secure user mode’.

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.14.14.6 State

SCPI Commands

```
MMEemory:STORe:STATe 1,
```

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: MMEemory:STORe:STATe
driver.massMemory.store.state.set(filename = '1')
```

This command saves the current instrument configuration in a *.dfl file.

param filename

String containing the path and name of the target file. The file extension is .dfl.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.store.state.clone()
```

Subgroups

6.14.14.6.1 Next

SCPI Commands

```
MMEMory:STORe:STATe:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: MMEMory:STORe:STATe:NEXT
driver.massMemory.store.state.next.set()
```

This command saves the current instrument configuration in a *.dfl file. The file name depends on the one you have set with method RsFswp.MassMemory.Store.State.set. This command adds a consecutive number to the file name.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: MMEMory:STORe:STATe:NEXT
driver.massMemory.store.state.next.set_with_opc()
```

This command saves the current instrument configuration in a *.dfl file. The file name depends on the one you have set with method RsFswp.MassMemory.Store.State.set. This command adds a consecutive number to the file name.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.14.14.7 Table

SCPI Commands

```
MMEMory:STORe<Store>:TABLE
```

class TableCls

Table commands group definition. 2 total commands, 1 Subgroups, 1 group commands

set(columns: StatisticType, filename: str, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:TABLE
driver.massMemory.store.table.set(columns = enums.StatisticType.ALL, filename =
↳ '1', store = repcap.Store.Default)
```

No command help available

param columns

No help available

param filename

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.massMemory.store.table.clone()
```

Subgroups

6.14.14.7.1 Limit

SCPI Commands

```
MMEemory:STORe<Store>:TABLE:LIMit
```

class LimitCls

Limit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(columns: StatisticType, filename: str, store=Store.Default) → None

```
# SCPI: MMEemory:STORe<n>:TABLE:LIMit
driver.massMemory.store.table.limit.set(columns = enums.StatisticType.ALL,
↳ filename = '1', store = repcap.Store.Default)
```

No command help available

param columns

No help available

param filename

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.8 Tfactor

SCPI Commands

`MMEMory:STORE<Store>:TFACtor`

class TfactorCls

Tfactor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class TfactorStruct

Response structure. Fields:

- Filename: str: No parameter help available
- Transd_Name: str: No parameter help available

get(*store=Store.Default*) → TfactorStruct

```
# SCPI: MMEMory:STORE<n>:TFACtor
value: TfactorStruct = driver.massMemory.store.tfactor.get(store = repcap.Store.
↪Default)
```

No command help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

return

structure: for return value, see the help for TfactorStruct structure arguments.

set(*filename: str, transd_name: str, store=Store.Default*) → None

```
# SCPI: MMEMory:STORE<n>:TFACtor
driver.massMemory.store.tfactor.set(filename = '1', transd_name = '1', store = ↪
↪repcap.Store.Default)
```

No command help available

param filename

No help available

param transd_name

No help available

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface 'Store')

6.14.14.9 Trace

SCPI Commands

```
MMEMory:STORe<Store>:TRACe 1,
```

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*trace: int, filename: str, store=Store.Default*) → None

```
# SCPI: MMEMory:STORe<n>:TRACe
driver.massMemory.store.trace.set(trace = 1, filename = '1', store = repcap.
↳Store.Default)
```

This command exports trace data from the specified window to an ASCII file. For details on the file format, see ‘Reference: ASCII file export format’. Secure User Mode In secure user mode, settings that are stored on the instrument are stored to volatile memory, which is restricted to 256 MB. Thus, a ‘memory limit reached’ error can occur although the hard disk indicates that storage space is still available. To store data permanently, select an external storage location such as a USB memory device. For details, see ‘Protecting data using the secure user mode’.

param trace

Number of the trace to be stored

param filename

String containing the path and name of the target file.

param store

optional repeated capability selector. Default value: Pos1 (settable in the interface ‘Store’)

6.14.14.10 TypePy

SCPI Commands

```
MMEMory:STORe:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → StoreType

```
# SCPI: MMEMory:STORe:TYPE
value: enums.StoreType = driver.massMemory.store.typePy.get()
```

This command defines whether the data from the entire instrument or only from the current channel is stored with the subsequent MMEM:STOR... command.

return

type_py: INSTRument | CHANnel INSTRument Stores data from the entire instrument.
CHANnel Stores data from an individual channel.

set(*type_py*: *StoreType*) → None

```
# SCPI: MMEMemory:STOR:TYPE
driver.massMemory.store.typePy.set(type_py = enums.StoreType.CHANnel)
```

This command defines whether the data from the entire instrument or only from the current channel is stored with the subsequent MME:STOR... command.

param type_py

INSTrument | CHANnel INSTrument Stores data from the entire instrument. CHAN-
nel Stores data from an individual channel.

6.15 Output<OutputConnector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.output.repcap_outputConnector_get()
driver.output.repcap_outputConnector_set(repcap.OutputConnector.Nr1)
```

class OutputCls

Output commands group definition. 14 total commands, 6 Subgroups, 0 group commands Repeated Capability:
OutputConnector, default value after init: OutputConnector.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.clone()
```

Subgroups

6.15.1 Ademod

class AdemodCls

Ademod commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.ademod.clone()
```


Subgroups

6.15.1.1 Online

class OnlineCls

Online commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.ademod.online.clone()
```

Subgroups

6.15.1.1.1 Af

class AfCls

Af commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.ademod.online.af.clone()
```

Subgroups

6.15.1.1.1.1 Cfrequency

SCPI Commands

```
OUTPut<OutputConnector>:ADEMod:ONLine:AF:CFrequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → float

```
# SCPI: OUTPut<up>:ADEMod[:ONLine]:AF[:CFrequency]
value: float = driver.output.ademod.online.af.cfrequency.get(outputConnector =
↳repcap.OutputConnector.Default)
```

This command defines the cutoff frequency for the AC highpass filter (for AC coupling only, see [SENSe:]ADEMod<n>:AF:COUPling) .

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

frequency: numeric value Range: 10 Hz to DemodBW/10 (= 300 kHz for active demodulation output) , Unit: HZ

set(frequency: float, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:ADEMod[:ONLine]:AF[:CFrequency]
driver.output.ademod.online.af.cfrequency.set(frequency = 1.0, outputConnector=
↳ repcap.OutputConnector.Default)
```

This command defines the cutoff frequency for the AC highpass filter (for AC coupling only, see [SENSe:]ADEMod<n>:AF:COUPling) .

param frequency

numeric value Range: 10 Hz to DemodBW/10 (= 300 kHz for active demodulation output) , Unit: HZ

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

6.15.1.1.2 Source**SCPI Commands**

```
OUTPut<OutputConnector>:ADEMod:ONLine:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → WindowName

```
# SCPI: OUTPut<up>:ADEMod[:ONLine]:SOURce
value: enums.WindowName = driver.output.ademod.online.source.
↳ get(outputConnector = repcap.OutputConnector.Default)
```

This command selects the result display whose results are output. Only active time domain results can be selected.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

return

window_name: (enum or string) string String containing the name of the window. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_query**. FOCus Dynamically switches to the currently selected window. If a window is selected that does not contain a time-domain result display, the selection is ignored and the previous setting is maintained.

set(window_name: WindowName, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:ADEMod[:ONLine]:SOURce
driver.output.ademod.online.source.set(window_name = enums.WindowName.FOCus,
↳ outputConnector = repcap.OutputConnector.Default)
```

This command selects the result display whose results are output. Only active time domain results can be selected.

param window_name

(enum or string) string String containing the name of the window. By default, the name of a window is the same as its index. To determine the name and index of all active windows, use the method **RsFswp.Layout.Catalog.Window.get_** query. FOCus Dynamically switches to the currently selected window. If a window is selected that does not contain a time-domain result display, the selection is ignored and the previous setting is maintained.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

6.15.1.1.3 State

SCPI Commands

```
OUTPut<OutputConnector>:ADEMod:ONLine:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → bool

```
# SCPI: OUTPut<up>:ADEMod[:ONLine][:STATe]
value: bool = driver.output.ademod.online.state.get(outputConnector = repcap.
↳OutputConnector.Default)
```

This command enables or disables online demodulation output to the IF/VIDEO/DEMODO output connector on the rear panel of the R&S FSWP.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:ADEMod[:ONLine][:STATe]
driver.output.ademod.online.state.set(state = False, outputConnector = repcap.
↳OutputConnector.Default)
```

This command enables or disables online demodulation output to the IF/VIDEO/DEMODO output connector on the rear panel of the R&S FSWP.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

6.15.2 Diq

class DiqCls

Diq commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.diq.clone()
```

Subgroups

6.15.2.1 State

SCPI Commands

```
OUTPut:DIQ:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: OUTPut:DIQ[:STATe]
value: bool = driver.output.diq.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: OUTPut:DIQ[:STATe]
driver.output.diq.state.set(state = False)
```

No command help available

param state
No help available

6.15.3 Ifreq

class IfreqCls

Ifreq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.ifreq.clone()
```

Subgroups

6.15.3.1 IfFrequency

SCPI Commands

```
OUTPut<OutputConnector>:IF:IFFrequency
```

class IfFrequencyCls

IfFrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → float

```
# SCPI: OUTPut<up>:IF:IFFrequency
value: float = driver.output.ifreq.ifFrequency.get(outputConnector = repcap.
↳OutputConnector.Default)
```

This command defines the frequency for the IF output of the R&S FSWP. The IF frequency of the signal is converted accordingly. This command is available in the time domain and if the IF/VIDEO/DEMODO output is configured for IF.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

return

frequency: Unit: HZ

set(frequency: float, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:IF:IFFrequency
driver.output.ifreq.ifFrequency.set(frequency = 1.0, outputConnector = repcap.
↳OutputConnector.Default)
```

This command defines the frequency for the IF output of the R&S FSWP. The IF frequency of the signal is converted accordingly. This command is available in the time domain and if the IF/VIDEO/DEMODO output is configured for IF.

param frequency

Unit: HZ

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

6.15.3.2 Source

SCPI Commands

`OUTPut<OutputConnector>:IF:SOURce`

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*outputConnector=OutputConnector.Default*) → Source

```
# SCPI: OUTPut<up>:IF[:SOURce]
value: enums.Source = driver.output.ifreq.source.get(outputConnector = repcap.
↳OutputConnector.Default)
```

Defines the type of signal available at one of the output connectors of the R&S FSWP. Note that you can use the audio frequency output only if the IF output source is 'Video'.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

return

source: IF The measured IF value is available at the IF/VIDEO/DEMOD output connector. VIDEO The displayed video signal (i.e. the filtered and detected IF signal, 200mV) is available at the IF/VIDEO/DEMOD output connector. This setting is required to provide demodulated audio frequencies at the output.

set(*source: Source, outputConnector=OutputConnector.Default*) → None

```
# SCPI: OUTPut<up>:IF[:SOURce]
driver.output.ifreq.source.set(source = enums.Source.AM, outputConnector =
↳repcap.OutputConnector.Default)
```

Defines the type of signal available at one of the output connectors of the R&S FSWP. Note that you can use the audio frequency output only if the IF output source is 'Video'.

param source

IF The measured IF value is available at the IF/VIDEO/DEMOD output connector. VIDEO The displayed video signal (i.e. the filtered and detected IF signal, 200mV) is available at the IF/VIDEO/DEMOD output connector. This setting is required to provide demodulated audio frequencies at the output.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

6.15.4 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.output.probe.repcap_probe_get()
driver.output.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.probe.clone()
```

Subgroups

6.15.4.1 Power

SCPI Commands

```
OUTPut<OutputConnector>:PROBe<Probe>:POWer
```

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, probe=Probe.Default) → bool

```
# SCPI: OUTPut<up>:PROBe<pb>[:POWer]
value: bool = driver.output.probe.power.get(outputConnector = repcap.
↳OutputConnector.Default, probe = repcap.Probe.Default)
```

No command help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

state: No help available

set(state: bool, outputConnector=OutputConnector.Default, probe=Probe.Default) → None

```
# SCPI: OUTPut<up>:PROBe<pb>[:POWer]
driver.output.probe.power.set(state = False, outputConnector = repcap.
↳OutputConnector.Default, probe = repcap.Probe.Default)
```

No command help available

param state

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Probe’)

6.15.5 Trigger<TriggerPort>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.output.trigger.repcap_triggerPort_get()
driver.output.trigger.repcap_triggerPort_set(repcap.TriggerPort.Nr1)
```

class TriggerCls

Trigger commands group definition. 5 total commands, 4 Subgroups, 0 group commands Repeated Capability: TriggerPort, default value after init: TriggerPort.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.trigger.clone()
```

Subgroups

6.15.5.1 Direction

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:DIRection
```

class DirectionCls

Direction commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → InOutDirection

```
# SCPI: OUTPut:TRIGger<2/3>:DIRection
value: enums.InOutDirection = driver.output.trigger.direction.get(triggerPort =
↳ repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

direction: INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

set(direction: InOutDirection, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<2/3>:DIRection
driver.output.trigger.direction.set(direction = enums.InOutDirection.INPut,
↳ triggerPort = repcap.TriggerPort.Default)
```

This command selects the trigger direction for trigger ports that serve as an input as well as an output.

param direction

INPut | OUTPut INPut Port works as an input. OUTPut Port works as an output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.15.5.2 Level

SCPI Commands

```
OUTPut:TRIGger<TriggerPort>:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → LowHigh

```
# SCPI: OUTPut:TRIGger<2/3>:LEVel
value: enums.LowHigh = driver.output.trigger.level.get(triggerPort = repcap.
↳ TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

level: HIGH 5 V LOW 0 V

set(level: LowHigh, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<2/3>:LEVel
driver.output.trigger.level.set(level = enums.LowHigh.HIGH, triggerPort =
↳ repcap.TriggerPort.Default)
```

This command defines the level of the (TTL compatible) signal generated at the trigger output. This command works only if you have selected a user-defined output with method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Otype.set.

param level

HIGH 5 V LOW 0 V

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.15.5.3 Otype**SCPI Commands**

```
OUTPut<OutputConnector>:TRIGger<TriggerPort>:OTYPE
```

class OtypeCls

Otype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → OutputType

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPE
value: enums.OutputType = driver.output.trigger.otype.get(outputConnector = ↵
↵repcap.OutputConnector.Default, triggerPort = repcap.TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output. Note: For offline AF or RF triggers, no output signal is provided.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

return

output_type: DEvice Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEfined Sends a user-defined trigger signal. For more information, see method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set.

set(output_type: OutputType, outputConnector=OutputConnector.Default, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut<up>:TRIGger<tp>:OTYPE
driver.output.trigger.otype.set(output_type = enums.OutputType.DEvice, ↵
↵outputConnector = repcap.OutputConnector.Default, triggerPort = repcap.
↵TriggerPort.Default)
```

This command selects the type of signal generated at the trigger output. Note: For offline AF or RF triggers, no output signal is provided.

param output_type

DEvice Sends a trigger signal when the R&S FSWP has triggered internally. TARMed Sends a trigger signal when the trigger is armed and ready for an external trigger event. UDEfined Sends a user-defined trigger signal. For more information, see method RsFswp.Applications.K30_NoiseFigure.Output.Trigger.Level.set.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Output')

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

6.15.5.4 Pulse**class PulseCls**

Pulse commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.trigger.pulse.clone()
```

Subgroups**6.15.5.4.1 Immediate****SCPI Commands**

```
OUTPut:TRIGger<TriggerPort>:PULSe:IMMediate
```

class ImmediateCls

Immediate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<2/3>:PULSe:IMMediate
driver.output.trigger.pulse.immediate.set(triggerPort = repcap.TriggerPort.
↳Default)
```

This command generates a pulse at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trigger')

set_with_opc(triggerPort=TriggerPort.Default, opc_timeout_ms: int = -1) → None

6.15.5.4.2 Length**SCPI Commands**

```
OUTPut:TRIGger<TriggerPort>:PULSe:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(triggerPort=TriggerPort.Default) → float

```
# SCPI: OUTPut:TRIGger<2/3>:PULSe:LENGth
value: float = driver.output.trigger.pulse.length.get(triggerPort = repcap.
↪TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

return

length: Pulse length in seconds. Unit: S

set(length: float, triggerPort=TriggerPort.Default) → None

```
# SCPI: OUTPut:TRIGger<2/3>:PULSe:LENGth
driver.output.trigger.pulse.length.set(length = 1.0, triggerPort = repcap.
↪TriggerPort.Default)
```

This command defines the length of the pulse generated at the trigger output.

param length

Pulse length in seconds. Unit: S

param triggerPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Trigger’)

6.15.6 Uport

class UportCls

Uport commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.output.uptport.clone()
```

Subgroups

6.15.6.1 State

SCPI Commands

```
OUTPut<OutputConnector>:UPORt:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → bool

```
# SCPI: OUTPut<up>:UPORt:StAtE
value: bool = driver.output.upt. state.get(outputConnector = repcap.
↳OutputConnector.Default)
```

This command toggles the control lines of the user ports for the AUX PORT connector. This 9-pole SUB-D male connector is located on the rear panel of the R&S FSWP.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

state: ON | OFF | 0 | 1 OFF | 0 User port is switched to INPut ON | 1 User port is switched to OUTPut

set(state: bool, outputConnector=OutputConnector.Default) → None

```
# SCPI: OUTPut<up>:UPORt:StAtE
driver.output.upt. state.set(state = False, outputConnector = repcap.
↳OutputConnector.Default)
```

This command toggles the control lines of the user ports for the AUX PORT connector. This 9-pole SUB-D male connector is located on the rear panel of the R&S FSWP.

param state

ON | OFF | 0 | 1 OFF | 0 User port is switched to INPut ON | 1 User port is switched to OUTPut

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

6.15.6.2 Value

SCPI Commands

```
OUTPut<OutputConnector>:UPORt:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(outputConnector=OutputConnector.Default) → str

```
# SCPI: OUTPut<up>:UPORt[:VALue]
value: str = driver.output.upt. value.get(outputConnector = repcap.
↳OutputConnector.Default)
```

This command sets the control lines of the user ports. The assignment of the pin numbers to the bits is as follows:

Table Header: Bit / 7 / 6 / 5 / 4 / 3 / 2 / 1 / 0

- Pin / N/A / N/A / 5 / 3 / 4 / 7 / 6 / 2

Bits 7 and 6 are not assigned to pins and must always be 0. The user port is written to with the given binary pattern. If the user port is programmed to input instead of output (see method `RsFswp.InputPy.Uport.State.set()`), the output value is temporarily stored.

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

value: bit values in hexadecimal format TTL type voltage levels (max. 5V) Range: #B00000000 to #B00111111

set(value: str, outputConnector=`OutputConnector.Default`) → None

```
# SCPI: OUTPut<up>:UPORt[:VALue]
driver.output.uptest.value.set(value = r1, outputConnector = repcap.
↳OutputConnector.Default)
```

This command sets the control lines of the user ports. The assignment of the pin numbers to the bits is as follows:

Table Header: Bit / 7 / 6 / 5 / 4 / 3 / 2 / 1 / 0

- Pin / N/A / N/A / 5 / 3 / 4 / 7 / 6 / 2

Bits 7 and 6 are not assigned to pins and must always be 0. The user port is written to with the given binary pattern. If the user port is programmed to input instead of output (see method `RsFswp.InputPy.Uport.State.set()`), the output value is temporarily stored.

param value

bit values in hexadecimal format TTL type voltage levels (max. 5V) Range: #B00000000 to #B00111111

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

6.16 Read

class ReadCls

Read commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.read.clone()
```

Subgroups

6.16.1 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.read.pmeter.repcap_powerMeter_get()
driver.read.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

SCPI Commands

```
READ:PMETer<PowerMeter>
```

class PmeterCls

Pmeter commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

get(*powerMeter=PowerMeter.Default*) → List[float]

```
# SCPI: READ:PMETer<p>
value: List[float] = driver.read.pmeter.get(powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.read.pmeter.clone()
```

6.17 Sense

class SenseCls

Sense commands group definition. 460 total commands, 26 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.clone()
```

Subgroups

6.17.1 Ademod

class AdemodCls

Ademod commands group definition. 42 total commands, 11 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.clone()
```

Subgroups

6.17.1.1 AdcPrefilter

SCPI Commands

```
SENSe:ADEMod:ADCPrefilter
```

class AdcPrefilterCls

AdcPrefilter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AdcPreFilterMode

```
# SCPI: [SENSe]:ADEMod:ADCPrefilter
value: enums.AdcPreFilterMode = driver.sense.ademod.adcPrefilter.get()
```

This command selects the bandwidth selection mode for the ADC prefilter.

return

mode: AUTO Selects the analog bandwidth based on the demodulation bandwidth.
WIDE Selects the largest possible analog bandwidth.

set(mode: AdcPreFilterMode) → None

```
# SCPI: [SENSe]:ADEMod:ADCPrefilter
driver.sense.ademod.adcPrefilter.set(mode = enums.AdcPreFilterMode.AUTO)
```

This command selects the bandwidth selection mode for the ADC prefilter.

param mode

AUTO Selects the analog bandwidth based on the demodulation bandwidth. WIDE
Selects the largest possible analog bandwidth.

6.17.1.2 Af

class AfCls

Af commands group definition. 6 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.af.clone()
```

Subgroups

6.17.1.2.1 Center

SCPI Commands

```
SENSe:ADEMod:AF:CENTer
```

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:AF:CENTer
value: float = driver.sense.ademod.af.center.get()
```

This command sets the center frequency for AF spectrum result display.

return
frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:ADEMod:AF:CENTer
driver.sense.ademod.af.center.set(frequency = 1.0)
```

This command sets the center frequency for AF spectrum result display.

param frequency
Unit: HZ

6.17.1.2.2 Coupling

SCPI Commands

```
SENSe:ADEMod:AF:COUPling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CouplingTypeA

```
# SCPI: [SENSe]:ADEMod:AF:COUpling
value: enums.CouplingTypeA = driver.sense.ademod.af.coupling.get()
```

This command selects the coupling of the AF path of the analyzer in the specified window.

return
coupling: AC | DC

set(coupling: CouplingTypeA) → None

```
# SCPI: [SENSe]:ADEMod:AF:COUpling
driver.sense.ademod.af.coupling.set(coupling = enums.CouplingTypeA.AC)
```

This command selects the coupling of the AF path of the analyzer in the specified window.

param coupling
AC | DC

6.17.1.2.3 Span

SCPI Commands

```
SENSe:ADEMod:AF:SPAN
```

class SpanCls

Span commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:AF:SPAN
value: float = driver.sense.ademod.af.span.get()
```

This command sets the span (around the center frequency) for AF spectrum result display. The span is limited to DBW/2 (see [SENSe:]BWIDth:DEMod) .

return
span: Unit: HZ

set(span: float) → None

```
# SCPI: [SENSe]:ADEMod:AF:SPAN
driver.sense.ademod.af.span.set(span = 1.0)
```

This command sets the span (around the center frequency) for AF spectrum result display. The span is limited to DBW/2 (see [SENSe:]BWIDth:DEMod) .

param span
Unit: HZ

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.af.span.clone()
```

Subgroups

6.17.1.2.3.1 Full

SCPI Commands

```
SENSe:ADEMod:AF:SPAN:FULL
```

class FullCls

Full commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADEMod:AF:SPAN:FULL
driver.sense.ademod.af.span.full.set()
```

This command sets the maximum span for AF spectrum result display. The maximum span corresponds to DBW/2 (see [SENSe:]BWIDth:DEMod).

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADEMod:AF:SPAN:FULL
driver.sense.ademod.af.span.full.set_with_opc()
```

This command sets the maximum span for AF spectrum result display. The maximum span corresponds to DBW/2 (see [SENSe:]BWIDth:DEMod).

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.1.2.4 Start

SCPI Commands

```
SENSe:ADEMod:AF:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:AF:START
value: float = driver.sense.ademod.af.start.get()
```

This command sets the start frequency for AF spectrum result display.

return
frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:ADEMod:AF:START
driver.sense.ademod.af.start.set(frequency = 1.0)
```

This command sets the start frequency for AF spectrum result display.

param frequency
Unit: HZ

6.17.1.2.5 Stop

SCPI Commands

```
SENSe:ADEMod:AF:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:AF:STOP
value: float = driver.sense.ademod.af.stop.get()
```

This command sets the stop frequency for AF spectrum result display.

return
frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:ADEMod:AF:STOP
driver.sense.ademod.af.stop.set(frequency = 1.0)
```

This command sets the stop frequency for AF spectrum result display.

param frequency
Unit: HZ

6.17.1.3 Am

class AmCls

Am commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.clone()
```

Subgroups

6.17.1.3.1 Absolute

class AbsoluteCls

Absolute commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.absolute.clone()
```

Subgroups

6.17.1.3.1.1 AfSpectrum

class AfSpectrumCls

AfSpectrum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.absolute.afSpectrum.clone()
```

Subgroups

6.17.1.3.1.2 Result

SCPI Commands

```
SENSe:ADEMod:AM:ABSolute:AFSPectrum:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute]:AFSPectrum:RESult
value: float = driver.sense.ademod.am.absolute.afSpectrum.result.get(trace_mode,
↪= enums.TraceModeB.AVERAGE)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.3.1.3 TypePy

SCPI Commands

```
SENSe:ADEMod:AM:ABSolute:AFSPpectrum:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute]:AFSPpectrum[:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.am.absolute.afSpectrum.
↳ typePy.get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute]:AFSPpectrum[:TYPE]
driver.sense.ademod.am.absolute.afSpectrum.typePy.set(trace_mode = [TraceModeA.
↳ AVERAge, TraceModeA.WRITe])
```

No command help available

param trace_mode

No help available

6.17.1.3.1.4 Tdomain

class TdomainCls

Tdomain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.absolute.tdomain.clone()
```

Subgroups

6.17.1.3.1.5 Result

SCPI Commands

```
SENSe:ADEMod:AM:ABSolute:TDOMain:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute][:TDOMain]:RESult
value: float = driver.sense.ademod.am.absolute.tdomain.result.get(trace_mode =
↳ enums.TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °

- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.3.1.6 TypePy

SCPI Commands

SENSe:ADEMod:AM:ABSolute:TDOMain:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute][:TDOMain][:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.am.absolute.tdomain.typePy.
↳get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:AM[:ABSolute][:TDOMain][:TYPE]
driver.sense.ademod.am.absolute.tdomain.typePy.set(trace_mode = [TraceModeA.
↳AVERAge, TraceModeA.WRITe])
```

No command help available

param trace_mode

No help available

6.17.1.3.2 Relative

class RelativeCls

Relative commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.relative.clone()
```

Subgroups

6.17.1.3.2.1 AfSpectrum

class AfSpectrumCls

AfSpectrum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.relative.afSpectrum.clone()
```

Subgroups

6.17.1.3.2.2 Result

SCPI Commands

```
SENSe:ADEMod:AM:RElative:AFSPpectrum:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:AM:RElative:AFSPpectrum:RESult
value: float = driver.sense.ademod.am.relative.afSpectrum.result.get(trace_mode_
↳ = enums.TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RElative[:TDOMain] / AM time domain / %
- AM:RElative:AFSpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz

- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.3.2.3 TypePy

SCPI Commands

SENSe:ADEMod:AM:RELative:AFSPpectrum:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:AM:RELative:AFSPpectrum[:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.am.relative.afSpectrum.
↳ typePy.get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:AM:RELative:AFSPpectrum[:TYPE]
driver.sense.ademod.am.relative.afSpectrum.typePy.set(trace_mode = [TraceModeA.
↳ AVERAge, TraceModeA.WRITe])
```

No command help available

param trace_mode

No help available

6.17.1.3.2.4 Tdomain

class TdomainCls

Tdomain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.am.relative.tdomain.clone()
```

Subgroups

6.17.1.3.2.5 Result

SCPI Commands

```
SENSe:ADEMod:AM:RElative:TDOMain:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:AM:RElative[:TDOMain]:RESult
value: float = driver.sense.ademod.am.relative.tdomain.result.get(trace_mode =
enums.TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RElative[:TDOMain] / AM time domain / %
- AM:RElative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERage | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENSE:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.3.2.6 TypePy**SCPI Commands**

`SENSe:ADEMod:AM:RELative:TDOMain:TYPE`

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:AM:RELative[:TDOMain][:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.am.relative.tdomain.typePy.
↳get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:AM:RELative[:TDOMain][:TYPE]
driver.sense.ademod.am.relative.tdomain.typePy.set(trace_mode = [TraceModeA.
↳AVERage, TraceModeA.WRITE])
```

No command help available

param trace_mode

No help available

6.17.1.4 Fm**class FmCls**

Fm commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.fm.clone()
```

Subgroups

6.17.1.4.1 AfSpectrum

class AfSpectrumCls

AfSpectrum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.fm.afSpectrum.clone()
```

Subgroups

6.17.1.4.1.1 Result

SCPI Commands

```
SENSe:ADEMod:FM:AFSPpectrum:REsult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:FM:AFSPpectrum:REsult
value: float = driver.sense.ademod.fm.afSpectrum.result.get(trace_mode = enums.
↳ TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

```
param trace_mode
    WRITe | AVERAge | MAXHold | MINHold

return
    trace_mode_result: The specified trace mode must be one of those configured by
    SENSE:ADEMod:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Other-
    wise a query error is generated.
```

6.17.1.4.1.2 TypePy

SCPI Commands

SENSe:ADEMod:FM:AFSPpectrum:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:FM:AFSPpectrum[:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.fm.afSpectrum.typePy.get()
```

No command help available

```
return
    trace_mode: No help available
```

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:FM:AFSPpectrum[:TYPE]
driver.sense.ademod.fm.afSpectrum.typePy.set(trace_mode = [TraceModeA.AVERAge,
↪TraceModeA.WRITe])
```

No command help available

```
param trace_mode
    No help available
```

6.17.1.4.2 Offset

SCPI Commands

SENSe:ADEMod:FM:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ResultTypeC

```
# SCPI: [SENSe]:ADEMod:FM:OFFSet
value: enums.ResultTypeC = driver.sense.ademod.fm.offset.get()
```

No command help available

return

result_type: No help available

set(result_type: ResultTypeC) → None

```
# SCPI: [SENSe]:ADEMod:FM:OFFSet
driver.sense.ademod.fm.offset.set(result_type = enums.ResultTypeC.AVERage)
```

No command help available

param result_type

No help available

6.17.1.4.3 Tdomain

class TdomainCls

Tdomain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.fm.tdomain.clone()
```

Subgroups

6.17.1.4.3.1 Result

SCPI Commands

SENSe:ADEMod:FM:TDOMain:RESult

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:FM[:TDOMain]:RESult
value: float = driver.sense.ademod.fm.tdomain.result.get(trace_mode = enums.
↳TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %

- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEMod:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.4.3.2 TypePy

SCPI Commands

SENSe:ADEMod:FM:TDOMain:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:FM[:TDOMain][:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.fm.tdomain.typePy.get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:FM[:TDOMain][:TYPE]
driver.sense.ademod.fm.tdomain.typePy.set(trace_mode = [TraceModeA.AVERAge,
↳TraceModeA.WRITe])
```

No command help available

param trace_mode

No help available

6.17.1.5 Mtime

SCPI Commands

```
SENSe:ADEMod:MTIME
```

class MtimeCls

Mtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:MTIME
value: float = driver.sense.ademod.mtime.get()
```

This command defines the measurement time for Analog Modulation Analysis.

return
time: Unit: S

set(time: float) → None

```
# SCPI: [SENSe]:ADEMod:MTIME
driver.sense.ademod.mtime.set(time = 1.0)
```

This command defines the measurement time for Analog Modulation Analysis.

param time
Unit: S

6.17.1.6 Pm

class PmCls

Pm commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.pm.clone()
```

Subgroups

6.17.1.6.1 AfSpectrum

class AfSpectrumCls

AfSpectrum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.pm.afSpectrum.clone()
```

Subgroups

6.17.1.6.1.1 Result

SCPI Commands

```
SENSe:ADEMod:PM:AFSPpectrum:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:PM:AFSPpectrum:RESult
value: float = driver.sense.ademod.pm.afSpectrum.result.get(trace_mode = enums.
    ↳TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERage | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.6.1.2 TypePy

SCPI Commands

```
SENSe:ADEMod:PM:AFSPpectrum:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:PM:AFSPpectrum[:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.pm.afSpectrum.typePy.get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:PM:AFSPpectrum[:TYPE]
driver.sense.ademod.pm.afSpectrum.typePy.set(trace_mode = [TraceModeA.AVERage,
↪TraceModeA.WRITE])
```

No command help available

param trace_mode

No help available

6.17.1.6.2 Rpoint

class RpointCls

Rpoint commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.pm.rpoint.clone()
```

Subgroups

6.17.1.6.2.1 X

SCPI Commands

```
SENSe:ADEMod:PM:RPOint:X
```

class XCls

X commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:PM:RPoint[:X]
value: float = driver.sense.ademod.pm.rpoint.x.get()
```

This command determines the position where the phase of the PM-demodulated signal is set to 0 rad. The maximum value depends on the measurement time selected in the instrument; this value is output in response to the query ADEM:PM:RPO:X? MAX.

return

time: 0 s to measurement time Unit: S

set(time: float) → None

```
# SCPI: [SENSe]:ADEMod:PM:RPoint[:X]
driver.sense.ademod.pm.rpoint.x.set(time = 1.0)
```

This command determines the position where the phase of the PM-demodulated signal is set to 0 rad. The maximum value depends on the measurement time selected in the instrument; this value is output in response to the query ADEM:PM:RPO:X? MAX.

param time

0 s to measurement time Unit: S

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.pm.rpoint.x.clone()
```

Subgroups

6.17.1.6.2.2 Mode

SCPI Commands

```
SENSe:ADEMod:PM:RPoint:X:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PmRpointMode

```
# SCPI: [SENSe]:ADEMod:PM:RPoint[:X]:MODE
value: enums.PmRpointMode = driver.sense.ademod.pm.rpoint.x.mode.get()
```

Defines how the reference position in time for 0 rad is determined.

return

mode: MANual | RIGHT MANual The time is defined using [SENSe:]ADEMod:PM:RPoint[:X]. RIGHT The time of the last measured value is used as the reference position. The time of the last measured value corresponds to the acquisition time, regarding the trigger event and trigger offset, if applicable. If the acquisition time or the trigger values are changed, the reference position is automatically adapted.

set(mode: PmRpointMode) → None

```
# SCPI: [SENSe]:ADEMod:PM:RPOint[:X]:MODE
driver.sense.ademod.pm.rpoint.x.mode.set(mode = enums.PmRpointMode.MANual)
```

Defines how the reference position in time for 0 rad is determined.

param mode

MANual | RIGHt MANual The time is defined using [SENSe:]ADEMod:PM:RPOint[:X]. RIGHt The time of the last measured value is used as the reference position. The time of the last measured value corresponds to the acquisition time, regarding the trigger event and trigger offset, if applicable. If the acquisition time or the trigger values are changed, the reference position is automatically adapted.

6.17.1.6.3 Tdomain

class TdomainCls

Tdomain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.pm.tdomain.clone()
```

Subgroups

6.17.1.6.3.1 Result

SCPI Commands

```
SENSe:ADEMod:PM:TDOMain:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:PM[:TDOMain]:RESult
value: float = driver.sense.ademod.pm.tdomain.result.get(trace_mode = enums.
↳TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSpectrum / AC-Video spectrum / V

- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz
- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENSE:ADEMod:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.6.3.2 TypePy**SCPI Commands**

SENSe:ADEMod:PM:TDOMain:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:PM[:TDOMain][:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.pm.tdomain.typePy.get()
```

No command help available

return

trace_mode: No help available

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:PM[:TDOMain][:TYPE]
driver.sense.ademod.pm.tdomain.typePy.set(trace_mode = [TraceModeA.AVERAge,
↪TraceModeA.WRITe])
```

No command help available

param trace_mode

No help available

6.17.1.7 Preset

class PresetCls

Preset commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.preset.clone()
```

Subgroups

6.17.1.7.1 Restore

SCPI Commands

```
SENSe:ADEMod:PRESet:REStore
```

class RestoreCls

Restore commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADEMod:PRESet:REStore
driver.sense.ademod.preset.restore.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADEMod:PRESet:REStore
driver.sense.ademod.preset.restore.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.1.7.2 Standard

SCPI Commands

```
SENSe:ADEMod:PRESet:STANdard
```

class StandardCls

Standard commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:ADEMod:PRESet[:STANdard]
value: str = driver.sense.ademod.preset.standard.get()
```

This command loads a measurement configuration. Standard definitions are stored in an xml file. The default directory for Analog Modulation Analysis standards is C:/R_S/INSTR/USER/predefined/AdemodPredefined.

return

standard: String containing the file name. If you have stored the file in a subdirectory of the directory mentioned above, you have to include the relative path to the file.

set(standard: str) → None

```
# SCPI: [SENSe]:ADEMod:PRESet[:STANdard]
driver.sense.ademod.preset.standard.set(standard = '1')
```

This command loads a measurement configuration. Standard definitions are stored in an xml file. The default directory for Analog Modulation Analysis standards is C:/R_S/INSTR/USER/predefined/AdemodPredefined.

param standard

String containing the file name. If you have stored the file in a subdirectory of the directory mentioned above, you have to include the relative path to the file.

6.17.1.7.3 Store

SCPI Commands

```
SENSe:ADEMod:PRESet:STORe
```

class StoreCls

Store commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:ADEMod:PRESet:STORe
value: str = driver.sense.ademod.preset.store.get()
```

This command saves the current Analog Modulation Analysis measurement configuration. Standard definitions are stored in an XML file. The default directory for Analog Modulation Analysis standards is C:/R_S/INSTR/USER/predefined/AdemodPredefined.

return

standard: String containing the file name. You can save the file in a subdirectory of the directory mentioned above. In that case, you have to include the relative path to the file.

set(standard: str) → None

```
# SCPI: [SENSe]:ADEMod:PRESet:STORe
driver.sense.ademod.preset.store.set(standard = '1')
```


This command saves the current Analog Modulation Analysis measurement configuration. Standard definitions are stored in an XML file. The default directory for Analog Modulation Analysis standards is C:/R_S/INSTR/USER/predefined/AdemodPredefined.

param standard

String containing the file name. You can save the file in a subdirectory of the directory mentioned above. In that case, you have to include the relative path to the file.

6.17.1.8 Set

SCPI Commands

```
SENSe:ADEMod:SET
```

class SetCls

Set commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetStruct

Structure for setting input parameters. Fields:

- **Sample_Rate:** float: numeric value The frequency at which measurement values are taken from the A/D-converter and stored in I/Q memory. Unit: HZ
- **Record_Length:** float: Number of samples to be stored in I/Q memory. Range: 1 to 400001 with AF filter or AF trigger active, 1 to 480001 with both AF filter and AF trigger deactive
- **Trigger_Source:** enums.TriggerSourceB: IMMEDIATE | EXTERNAL | EXT2 | EXT3 | IFPower | RFPower | AF | AM | AMRelative | FM | PM Note: After selecting IF Power, the trigger threshold can be set with the [CMDLINK: TRIGGER[:SEQUENCE]:LEVEL:IFPower CMDLINK] command.
- **Trigger_Slope:** enums.SlopeType: POSITIVE | NEGATIVE Used slope of the trigger signal. The value indicated here will be ignored for trigger source = IMMEDIATE.
- **Offset_Samples:** float: Number of samples to be used as an offset to the trigger signal. The value indicated here is ignored for trigger source = 'IMMEDIATE'.
- **No_Of_Meas:** float: Number of repetitions of the measurement to be executed. The value indicated here is especially necessary for the average/maxhold/minhold function. Range: 0 to 32767

get() → SetStruct

```
# SCPI: [SENSe]:ADEMod:SET
value: SetStruct = driver.sense.ademod.set.get()
```

This command configures the analog demodulator of the instrument.

return

structure: for return value, see the help for SetStruct structure arguments.

set(structure: SetStruct) → None

```
# SCPI: [SENSe]:ADEMod:SET
structure = driver.sense.ademod.set.SetStruct()
structure.Sample_Rate: float = 1.0
structure.Record_Length: float = 1.0
structure.Trigger_Source: enums.TriggerSourceB = enums.TriggerSourceB.ACVideo
structure.Trigger_Slope: enums.SlopeType = enums.SlopeType.NEGATIVE
```

(continues on next page)

(continued from previous page)

```
structure.Offset_Samples: float = 1.0
structure.No_Of_Meas: float = 1.0
driver.sense.ademod.set.set(structure)
```

This command configures the analog demodulator of the instrument.

param structure

for set value, see the help for SetStruct structure arguments.

6.17.1.9 Spectrum

class SpectrumCls

Spectrum commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.spectrum.clone()
```

Subgroups

6.17.1.9.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.spectrum.bandwidth.clone()
```

Subgroups

6.17.1.9.1.1 Resolution

SCPI Commands

```
SENSe:ADEMod:SPECTrum:BWIDth:RESolution
```

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:SPECTrum:BWIDth[:RESolution]
value: float = driver.sense.ademod.spectrum.bandwidth.resolution.get()
```

Defines the resolution bandwidth for data acquisition. From the specified RBW and the demodulation span set by [SENSe:]ADEMod:SPECTrum:SPAN[:MAXimum] or [SENSe:]BWIDth:DEMod, the required measurement time is calculated. If the available measurement time is not sufficient for the given bandwidth, the measurement time is set to its maximum and the resolution bandwidth is increased to the resulting bandwidth. This command is identical to [SENSe:]BANDwidth[:RESolution].

return

bandwidth: refer to data sheet Unit: HZ

set(bandwidth: float) → None

```
# SCPI: [SENSe]:ADEMod:SPECTrum:BWIDth[:RESolution]
driver.sense.ademod.spectrum.bandwidth.resolution.set(bandwidth = 1.0)
```

Defines the resolution bandwidth for data acquisition. From the specified RBW and the demodulation span set by [SENSe:]ADEMod:SPECTrum:SPAN[:MAXimum] or [SENSe:]BWIDth:DEMod, the required measurement time is calculated. If the available measurement time is not sufficient for the given bandwidth, the measurement time is set to its maximum and the resolution bandwidth is increased to the resulting bandwidth. This command is identical to [SENSe:]BANDwidth[:RESolution].

param bandwidth

refer to data sheet Unit: HZ

6.17.1.9.2 Result

SCPI Commands

```
SENSe:ADEMod:SPECTrum:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_mode: TraceModeB) → float

```
# SCPI: [SENSe]:ADEMod:SPECTrum:RESult
value: float = driver.sense.ademod.spectrum.result.get(trace_mode = enums.
↳ TraceModeB.AVERage)
```

This command reads the result data of the evaluated signal in the specified trace mode. The data format of the output data block is defined by the FORMat command (see method RsFswp.FormatPy.Data.set) . The trace results are configured for a specific evaluation. The following table indicates which command syntax refers to which evaluation method, as well as the output unit of the results.

Table Header: Command syntax / Evaluation method / Output unit

- ACV[:TDOMain] / AC-Video time domain / V
- ACV:AFSPpectrum / AC-Video spectrum / V
- AM[:ABSolute][:TDOMain] / RF time domain / dBm
- AM:RELative[:TDOMain] / AM time domain / %
- AM:RELative:AFSPpectrum / AM spectrum / %
- FM[:TDOMain] / FM time domain / kHz
- FM:AFSPpectrum / FM spectrum / kHz

- PM[:TDOMain] / PM time domain / rad or °
- PM:AFSPpectrum / PM spectrum / rad or °
- SPECTrum / RF spectrum / dBm (logarithmic display) or V (linear display) .

param trace_mode

WRITe | AVERAge | MAXHold | MINHold

return

trace_mode_result: The specified trace mode must be one of those configured by SENS:ADEM:Evaluation:TYPE, see [SENSe:]ADEMod:SPECTrum[:TYPE]. Otherwise a query error is generated.

6.17.1.9.3 Span

class SpanCls

Span commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.spectrum.span.clone()
```

Subgroups

6.17.1.9.3.1 Maximum

SCPI Commands

```
SENSe:ADEMod:SPECTrum:SPAN:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:SPECTrum:SPAN[:MAXimum]
value: float = driver.sense.ademod.spectrum.span.maximum.get()
```

Sets the DBW to the specified value and the span (around the center frequency) of the RF data to be evaluated to its new maximum (the demodulation bandwidth) .

return

freq_range: Unit: Hz

set(freq_range: float) → None

```
# SCPI: [SENSe]:ADEMod:SPECTrum:SPAN[:MAXimum]
driver.sense.ademod.spectrum.span.maximum.set(freq_range = 1.0)
```

Sets the DBW to the specified value and the span (around the center frequency) of the RF data to be evaluated to its new maximum (the demodulation bandwidth) .

param freq_range
Unit: Hz

6.17.1.9.3.2 Zoom

SCPI Commands

```
SENSe:ADEMod:SPECTrum:SPAN:ZOOM
```

class ZoomCls

Zoom commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:SPECTrum:SPAN:ZOOM
value: float = driver.sense.ademod.spectrum.span.zoom.get()
```

This command sets the span (around the center frequency) for RF spectrum result display. The span is limited to the demodulation bandwidth (see [SENSe:]BWIDth:DEMod) .

return
span: Unit: HZ

set(span: float) → None

```
# SCPI: [SENSe]:ADEMod:SPECTrum:SPAN:ZOOM
driver.sense.ademod.spectrum.span.zoom.set(span = 1.0)
```

This command sets the span (around the center frequency) for RF spectrum result display. The span is limited to the demodulation bandwidth (see [SENSe:]BWIDth:DEMod) .

param span
Unit: HZ

6.17.1.9.4 TypePy

SCPI Commands

```
SENSe:ADEMod:SPECTrum:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[TraceModeA]

```
# SCPI: [SENSe]:ADEMod:SPECTrum[:TYPE]
value: List[enums.TraceModeA] = driver.sense.ademod.spectrum.typePy.get()
```

This command selects the trace modes of the evaluated signal to be measured simultaneously. For each of the six available traces a mode can be defined. For details on trace modes see ‘Mode’. The trace modes are configured identically for all windows with a specific evaluation. The following table indicates which command syntax refers to which evaluation method.

Table Header: Command syntax / Evaluation method

- AM[:ABSolute][:TDOMain] / RF time domain
- AM:RELative[:TDOMain] / AM time domain
- AM:RELative:AFSPpectrum / AM spectrum (relative)
- FM[:TDOMain] / FM time domain
- FM:AFSPpectrum / FM spectrum
- PM[:TDOMain] / PM time domain
- PM:AFSPpectrum / PM spectrum
- SPECTrum / RF spectrum

return

trace_mode: WRITE | AVERage | MAXHold | MINHold | VIEW | OFF WRITE Overwrite mode: the trace is overwritten by each sweep. This is the default setting. AVERage The average is formed over several sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is greater than the previous one. MINHold The minimum value is determined from several measurements and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is lower than the previous one. VIEW The current contents of the trace memory are frozen and displayed. OFF Hides the selected trace.

set(trace_mode: List[TraceModeA]) → None

```
# SCPI: [SENSe]:ADEMod:SPECTrum[:TYPE]
driver.sense.ademod.spectrum.typePy.set(trace_mode = [TraceModeA.AVERage,
↪TraceModeA.WRITE])
```

This command selects the trace modes of the evaluated signal to be measured simultaneously. For each of the six available traces a mode can be defined. For details on trace modes see 'Mode'. The trace modes are configured identically for all windows with a specific evaluation. The following table indicates which command syntax refers to which evaluation method.

Table Header: Command syntax / Evaluation method

- AM[:ABSolute][:TDOMain] / RF time domain
- AM:RELative[:TDOMain] / AM time domain
- AM:RELative:AFSPpectrum / AM spectrum (relative)
- FM[:TDOMain] / FM time domain
- FM:AFSPpectrum / FM spectrum
- PM[:TDOMain] / PM time domain
- PM:AFSPpectrum / PM spectrum
- SPECTrum / RF spectrum

param trace_mode

WRITE | AVERage | MAXHold | MINHold | VIEW | OFF WRITE Overwrite mode: the trace is overwritten by each sweep. This is the default setting. AVERage The average is formed over several sweeps. MAXHold The maximum value is determined over several sweeps and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is greater than the previous one. MINHold The minimum

value is determined from several measurements and displayed. The R&S FSWP saves the sweep result in the trace memory only if the new value is lower than the previous one. VIEW The current contents of the trace memory are frozen and displayed. OFF Hides the selected trace.

6.17.1.10 Squelch

class SquelchCls

Squelch commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.squelch.clone()
```

Subgroups

6.17.1.10.1 Level

SCPI Commands

```
SENSe:ADEMod:SQUelch:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:SQUelch:LEVel
value: float = driver.sense.ademod.squelch.level.get()
```

This command defines the level threshold below which the demodulated data is set to 0 if squelching is enabled (see [SENSe:]ADEMod:SQUelch[:STATe]) .

return

threshold: numeric value The absolute threshold level Range: -150 dBm to 30 dBm

set(threshold: float) → None

```
# SCPI: [SENSe]:ADEMod:SQUelch:LEVel
driver.sense.ademod.squelch.level.set(threshold = 1.0)
```

This command defines the level threshold below which the demodulated data is set to 0 if squelching is enabled (see [SENSe:]ADEMod:SQUelch[:STATe]) .

param threshold

numeric value The absolute threshold level Range: -150 dBm to 30 dBm

6.17.1.10.2 State

SCPI Commands

`SENSe:ADEMod:SQUelch:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ADEMod:SQUelch[:STATe]
value: bool = driver.sense.ademod.squelch.state.get()
```

This command activates the squelch function, i.e. if the signal falls below a defined threshold (see [SENSe:]ADEMod:SQUelch:LEVeL), the demodulated data is automatically set to 0.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:ADEMod:SQUelch[:STATe]
driver.sense.ademod.squelch.state.set(state = False)
```

This command activates the squelch function, i.e. if the signal falls below a defined threshold (see [SENSe:]ADEMod:SQUelch:LEVeL), the demodulated data is automatically set to 0.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.17.1.11 Zoom

class ZoomCls

Zoom commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.zoom.clone()
```

Subgroups

6.17.1.11.1 Length

SCPI Commands

`SENSe:ADEMod:ZOOM:LENGth`

class LengthCls

Length commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:ZOOM:LENGth
value: float = driver.sense.ademod.zoom.length.get()
```

The command allows you to define the length of the time domain zoom area for the analog-demodulated measurement data in the specified window manually. If the length is defined manually using this command, the zoom mode is also set to manual.

return

length: Unit: S Length of the zoom area in seconds.

set(length: float) → None

```
# SCPI: [SENSe]:ADEMod:ZOOM:LENGth
driver.sense.ademod.zoom.length.set(length = 1.0)
```

The command allows you to define the length of the time domain zoom area for the analog-demodulated measurement data in the specified window manually. If the length is defined manually using this command, the zoom mode is also set to manual.

param length

Unit: S Length of the zoom area in seconds.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ademod.zoom.length.clone()
```

Subgroups**6.17.1.11.1.1 Mode****SCPI Commands**

```
SENSe:ADEMod:ZOOM:LENGth:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:ADEMod:ZOOM:LENGth:MODE
value: enums.AutoManualMode = driver.sense.ademod.zoom.length.mode.get()
```

The command defines whether the length of the zoom area for the analog-demodulated measurement data is defined automatically or manually in the specified window.

return

mode: AUTO | MAN AUTO (Default:) The number of sweep points is

used as the zoom length. MAN The zoom length is defined manually using [SENSe:]ADEModn:ZOOM:LENGth.

set(*mode: AutoManualMode*) → None

```
# SCPI: [SENSe]:ADEMod:ZOOM:LENGth:MODE
driver.sense.ademod.zoom.length.mode.set(mode = enums.AutoManualMode.AUTO)
```

The command defines whether the length of the zoom area for the analog-demodulated measurement data is defined automatically or manually in the specified window.

param mode

AUTO | MAN AUTO (Default:) The number of sweep points is used as the zoom length. MAN The zoom length is defined manually using [SENSe:]ADEModn:ZOOM:LENGth.

6.17.1.11.2 Start

SCPI Commands

```
SENSe:ADEMod:ZOOM:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADEMod:ZOOM:START
value: float = driver.sense.ademod.zoom.start.get()
```

The command selects the start time for the zoomed display of analog-demodulated measurements in the specified window. The maximum value depends on the measurement time, which is set and can be queried with the [SENSe:]ADEMod:MTIME command. If the zoom function is enabled, the defined number of sweep points are displayed from the start time specified with this command.

return

time: Range: 0 s to (measurement time – zoom length) , Unit: S

set(*time: float*) → None

```
# SCPI: [SENSe]:ADEMod:ZOOM:START
driver.sense.ademod.zoom.start.set(time = 1.0)
```

The command selects the start time for the zoomed display of analog-demodulated measurements in the specified window. The maximum value depends on the measurement time, which is set and can be queried with the [SENSe:]ADEMod:MTIME command. If the zoom function is enabled, the defined number of sweep points are displayed from the start time specified with this command.

param time

Range: 0 s to (measurement time – zoom length) , Unit: S

6.17.1.11.3 State

SCPI Commands

```
SENSe:ADEMod:ZOOM:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ADEMod:ZOOM[:STATe]
value: bool = driver.sense.ademod.zoom.state.get()
```

The command enables or disables the time domain zoom function for the analog-demodulated measurement data in the specified window. If the zoom function is enabled, the defined number of sweep points are displayed from the start time specified with [SENSe:]ADEMod<n>:ZOOM:START. If the zoom function is disabled, data reduction is used to adapt the measurement points to the number of points available on the display.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:ADEMod:ZOOM[:STATe]
driver.sense.ademod.zoom.state.set(state = False)
```

The command enables or disables the time domain zoom function for the analog-demodulated measurement data in the specified window. If the zoom function is enabled, the defined number of sweep points are displayed from the start time specified with [SENSe:]ADEMod<n>:ZOOM:START. If the zoom function is disabled, data reduction is used to adapt the measurement points to the number of points available on the display.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.17.2 Adjust

class AdjustCls

Adjust commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.clone()
```

Subgroups

6.17.2.1 All

SCPI Commands

SENSe:ADJust:ALL

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

<pre># SCPI: [SENSe]:ADJust:ALL driver.sense.adjust.all.set()</pre>

This command initiates a measurement to determine and set the ideal settings for the current task automatically (only once for the current measurement) . This includes:

INTRO_CMD_HELP: Prerequisites for this command

- Center frequency
- Reference level
- Scaling

set_with_opc(opc_timeout_ms: int = -1) → None

<pre># SCPI: [SENSe]:ADJust:ALL driver.sense.adjust.all.set_with_opc()</pre>
--

This command initiates a measurement to determine and set the ideal settings for the current task automatically (only once for the current measurement) . This includes:

INTRO_CMD_HELP: Prerequisites for this command

- Center frequency
- Reference level
- Scaling

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.2.2 Configure

class ConfigureCls

Configure commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.clone()
```

Subgroups

6.17.2.2.1 Duration

SCPI Commands

```
SENSe:ADJust:CONFigure:DURation
```

class DurationCls

Duration commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:DURation
value: float = driver.sense.adjust.configure.duration.get()
```

No command help available

```
return
    duration: No help available
```

set(duration: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:DURation
driver.sense.adjust.configure.duration.set(duration = 1.0)
```

No command help available

```
param duration
    No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.duration.clone()
```

Subgroups

6.17.2.2.1.1 Mode

SCPI Commands

`SENSe:ADJust:CONFigure:DURation:MODE`

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:ADJust:CONFigure:DURation:MODE
value: enums.AutoManualMode = driver.sense.adjust.configure.duration.mode.get()
```

No command help available

return

mode: No help available

set(mode: AutoManualMode) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:DURation:MODE
driver.sense.adjust.configure.duration.mode.set(mode = enums.AutoManualMode.
↳AUTO)
```

No command help available

param mode

No help available

6.17.2.2.2 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.frequency.clone()
```

Subgroups

6.17.2.2.2.1 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.frequency.limit.clone()
```

Subgroups

6.17.2.2.2.2 High

SCPI Commands

```
SENSe:ADJust:CONFigure:FREQuency:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:FREQuency:LIMit:HIGH
value: float = driver.sense.adjust.configure.frequency.limit.high.get()
```

This command defines the upper limit of the frequency search range.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

return

frequency: numeric value Range: See data sheet , Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:FREQuency:LIMit:HIGH
driver.sense.adjust.configure.frequency.limit.high.set(frequency = 1.0)
```

This command defines the upper limit of the frequency search range.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

param frequency

numeric value Range: See data sheet , Unit: Hz

6.17.2.2.2.3 Low

SCPI Commands

```
SENSe:ADJust:CONFigure:FREQuency:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:FREQuency:LIMit:LOW
value: float = driver.sense.adjust.configure.frequency.limit.low.get()
```

This command defines the lower limit of the frequency search range.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe]:ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

return

frequency: numeric value Range: See data sheet , Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:FREQuency:LIMit:LOW
driver.sense.adjust.configure.frequency.limit.low.set(frequency = 1.0)
```

This command defines the lower limit of the frequency search range.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe]:ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

param frequency

numeric value Range: See data sheet , Unit: Hz

6.17.2.2.3 Hysteresis

class HysteresisCls

Hysteresis commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.hysteresis.clone()
```

Subgroups

6.17.2.2.3.1 Lower

SCPI Commands

```
SENSe:ADJust:CONFigure:HYSTeresis:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float


```
# SCPI: [SENSe]:ADJust:CONFigure:HYSTeresis:LOWer
value: float = driver.sense.adjust.configure.hysteresis.lower.get()
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines a lower threshold the signal must fall below (compared to the last measurement) before the reference level is adapted automatically.

return

threshold: Range: 0 dB to 200 dB, Unit: dB

set(threshold: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:HYSTeresis:LOWer
driver.sense.adjust.configure.hysteresis.lower.set(threshold = 1.0)
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines a lower threshold the signal must fall below (compared to the last measurement) before the reference level is adapted automatically.

param threshold

Range: 0 dB to 200 dB, Unit: dB

6.17.2.2.3.2 Upper

SCPI Commands

```
SENSe:ADJust:CONFigure:HYSTeresis:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:HYSTeresis:UPPer
value: float = driver.sense.adjust.configure.hysteresis.upper.get()
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines an upper threshold the signal must exceed (compared to the last measurement) before the reference level is adapted automatically.

return

threshold: Range: 0 dB to 200 dB, Unit: dB

set(threshold: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:HYSTeresis:UPPer
driver.sense.adjust.configure.hysteresis.upper.set(threshold = 1.0)
```

When the reference level is adjusted automatically using the [SENSe:]ADJust:LEVel command, the internal attenuators and the preamplifier are also adjusted. To avoid frequent adaptation due to small changes in the input signal, you can define a hysteresis. This setting defines an upper threshold the signal must exceed (compared to the last measurement) before the reference level is adapted automatically.

param threshold

Range: 0 dB to 200 dB, Unit: dB

6.17.2.2.4 Level**class LevelCls**

Level commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.level.clone()
```

Subgroups**6.17.2.2.4.1 Duration****SCPI Commands**

```
SENSe:ADJust:CONFigure:LEVel:DURation
```

class DurationCls

Duration commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:LEVel:DURation
value: float = driver.sense.adjust.configure.level.duration.get()
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command defines the length of the measurement if [SENSe:]ADJust:CONFigure:LEVel:DURation:MODE is set to MANual.

return

duration: Numeric value in seconds Range: 0.001 to 16000.0, Unit: s

set(duration: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:LEVel:DURation
driver.sense.adjust.configure.level.duration.set(duration = 1.0)
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command defines the length of the measurement if [SENSe:]ADJust:CONFigure:LEVel:DURation:MODE is set to MANual.

param duration

Numeric value in seconds Range: 0.001 to 16000.0, Unit: s

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.configure.level.duration.clone()
```

Subgroups

6.17.2.2.4.2 Mode

SCPI Commands

```
SENSe:ADJust:CONFigure:LEVel:DURation:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:ADJust:CONFigure[:LEVel]:DURation:MODE
value: enums.AutoManualMode = driver.sense.adjust.configure.level.duration.mode.
↳ get()
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command selects the way the R&S FSWP determines the length of the measurement .

return

mode: AUTO The R&S FSWP determines the measurement length automatically according to the current input data. MANual The R&S FSWP uses the measurement length defined by [SENSe:]ADJust:CONFigure:LEVel:DURation.

set(mode: AutoManualMode) → None

```
# SCPI: [SENSe]:ADJust:CONFigure[:LEVel]:DURation:MODE
driver.sense.adjust.configure.level.duration.mode.set(mode = enums.
↳ AutoManualMode.AUTO)
```

To determine the ideal reference level, the R&S FSWP performs a measurement on the current input data. This command selects the way the R&S FSWP determines the length of the measurement .

param mode

AUTO The R&S FSWP determines the measurement length automatically according to the current input data. MANual The R&S FSWP uses the measurement length defined by [SENSe:]ADJust:CONFigure:LEVel:DURation.

6.17.2.2.4.3 Threshold

SCPI Commands

`SENSe:ADJust:CONFigure:LEVel:THReshold`

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ADJust:CONFigure:LEVel:THReshold
value: float = driver.sense.adjust.configure.level.threshold.get()
```

This command defines the threshold of the signal search.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

return

level: numeric value Unit: dBm

set(level: float) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:LEVel:THReshold
driver.sense.adjust.configure.level.threshold.set(level = 1.0)
```

This command defines the threshold of the signal search.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the automatic signal search with [SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]

param level

numeric value Unit: dBm

6.17.2.2.5 Trigger

SCPI Commands

`SENSe:ADJust:CONFigure:TRIGger`

class TriggerCls

Trigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ADJust:CONFigure:TRIGger
value: bool = driver.sense.adjust.configure.trigger.get()
```

Defines the behavior of the measurement when adjusting a setting automatically (using SENS:ADJ:LEV ON, for example) . See ‘Adjusting settings automatically during triggered measurements’.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:ADJust:CONFigure:TRIGger
driver.sense.adjust.configure.trigger.set(state = False)
```

Defines the behavior of the measurement when adjusting a setting automatically (using SENS:ADJ:LEV ON, for example) . See ‘Adjusting settings automatically during triggered measurements’.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.17.2.3 Frequency

SCPI Commands

```
SENSe:ADJust:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:ADJust:FREQuency
driver.sense.adjust.frequency.set()
```

This command sets the center frequency to the frequency with the highest signal level in the current frequency range.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:ADJust:FREQuency
driver.sense.adjust.frequency.set_with_opc()
```

This command sets the center frequency to the frequency with the highest signal level in the current frequency range.

Same as set, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.2.4 Level

SCPI Commands

`SENSe:ADJust:LEVel`

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*opc_timeout_ms: int = -1*) → None

```
# SCPI: [SENSe]:ADJust:LEVel
driver.sense.adjust.level.set()
```

Initiates a single (internal) measurement that evaluates and sets the ideal reference level for the current input data and measurement settings. Thus, the settings of the RF attenuation and the reference level are optimized for the signal level. The R&S FSWP is not overloaded and the dynamic range is not limited by an S/N ratio that is too small.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.2.5 Scale

class ScaleCls

Scale commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.scale.clone()
```

Subgroups

6.17.2.5.1 Y

class YCls

Y commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.scale.y.clone()
```

Subgroups

6.17.2.5.1.1 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.adjust.scale.y.auto.clone()
```

Subgroups

6.17.2.5.1.2 Continuous

SCPI Commands

```
SENSe:ADJust:SCALe:Y:AUTO:CONTInuous
```

class ContinuousCls

Continuous commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ADJust:SCALe[:Y]:AUTO[:CONTInuous]
value: bool = driver.sense.adjust.scale.y.auto.continuous.get()
```

Activates automatic scaling of the y-axis in all diagrams according to the current measurement results. Currently auto-scaling is only available for AF measurements. RF power and RF spectrum measurements are not affected by the auto-scaling.

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool) → None

```
# SCPI: [SENSe]:ADJust:SCALe[:Y]:AUTO[:CONTInuous]
driver.sense.adjust.scale.y.auto.continuous.set(state = False)
```

Activates automatic scaling of the y-axis in all diagrams according to the current measurement results. Currently auto-scaling is only available for AF measurements. RF power and RF spectrum measurements are not affected by the auto-scaling.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

6.17.3 Average

class AverageCls

Average commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.average.clone()
```

Subgroups

6.17.3.1 Count

SCPI Commands

```
SENSe:AVERage:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:AVERage:COUNT
value: float = driver.sense.average.count.get()
```

This command defines the number of measurements that the application uses to average traces. In case of continuous sweep mode, the application calculates the moving average over the average count. In case of single sweep mode, the application stops the measurement and calculates the average after the average count has been reached.

return

average_count: If you set an average count of 0 or 1, the application performs one single measurement in single sweep mode. In continuous sweep mode, if the average count is set to 0, a moving average over 10 measurements is performed. Range: 0 to 200000

set(average_count: float) → None

```
# SCPI: [SENSe]:AVERage:COUNT
driver.sense.average.count.set(average_count = 1.0)
```

This command defines the number of measurements that the application uses to average traces. In case of continuous sweep mode, the application calculates the moving average over the average count. In case of single sweep mode, the application stops the measurement and calculates the average after the average count has been reached.

param average_count

If you set an average count of 0 or 1, the application performs one single measurement in single sweep mode. In continuous sweep mode, if the average count is set to 0, a moving average over 10 measurements is performed. Range: 0 to 200000

6.17.3.2 State<Status>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.average.state.repcap_status_get()
driver.sense.average.state.repcap_status_set(repcap.Status.Nr1)
```

SCPI Commands

```
SENSe:AVERage:STATe<Status>
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Status, default value after init: Status.Nr1

get(*status=Status.Default*) → bool

```
# SCPI: [SENSe]:AVERage:STATe<t>
value: bool = driver.sense.average.state.get(status = repcap.Status.Default)
```

This command turns averaging of the I/Q data on and off. Before you can use the command you have to turn the I/Q data acquisition on with method RsFswp.Applications.IqAnalyzer.Trace.Iq.State.set. If averaging is on, the maximum amount of I/Q data that can be recorded is 512kS (524288 samples) .

param status

optional repeated capability selector. Default value: Nr1 (settable in the interface 'State')

return

average_mode: No help available

set(*average_mode: bool, status=Status.Default*) → None

```
# SCPI: [SENSe]:AVERage:STATe<t>
driver.sense.average.state.set(average_mode = False, status = repcap.Status.
↪Default)
```

This command turns averaging of the I/Q data on and off. Before you can use the command you have to turn the I/Q data acquisition on with method RsFswp.Applications.IqAnalyzer.Trace.Iq.State.set. If averaging is on, the maximum amount of I/Q data that can be recorded is 512kS (524288 samples) .

param average_mode

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param status

optional repeated capability selector. Default value: Nr1 (settable in the interface 'State')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.average.state.clone()
```

6.17.3.3 TypePy

SCPI Commands

```
SENSe:AVERage:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AverageModeB

```
# SCPI: [SENSe]:AVERage:TYPE
value: enums.AverageModeB = driver.sense.average.typePy.get()
```

No command help available

return

mode: No help available

set(mode: AverageModeB) → None

```
# SCPI: [SENSe]:AVERage:TYPE
driver.sense.average.typePy.set(mode = enums.AverageModeB.LINEar)
```

No command help available

param mode

No help available

6.17.4 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 11 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.bandwidth.clone()
```

Subgroups

6.17.4.1 Demod

SCPI Commands

```
SENSe:BWIDth:DEMod
```

class DemodCls

Demod commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:BWIDth:DEMod
value: float = driver.sense.bandwidth.demod.get()
```

This command sets the bandwidth for Analog Modulation Analysis. Depending on the selected demodulation bandwidth, the instrument selects the required sample rate. This command is identical to SENS:ADEM:BAND:DEM.

return
bandwidth: Unit: HZ

set(bandwidth: float) → None

```
# SCPI: [SENSe]:BWIDth:DEMod
driver.sense.bandwidth.demod.set(bandwidth = 1.0)
```

This command sets the bandwidth for Analog Modulation Analysis. Depending on the selected demodulation bandwidth, the instrument selects the required sample rate. This command is identical to SENS:ADEM:BAND:DEM.

param bandwidth
Unit: HZ

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.bandwidth.demod.clone()
```

Subgroups

6.17.4.1.1 TypePy

SCPI Commands

```
SENSe:BWIDth:DEMod:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FilterTypeA

```
# SCPI: [SENSe]:BWIDth:DEMod:TYPE
value: enums.FilterTypeA = driver.sense.bandwidth.demod.typePy.get()
```

This command defines the type of demodulation filter to be used. This command is identical to SENS:ADEM:BAND:DEM:TYPE:

return
filter_type: FLAT Standard flat demodulation filter GAUSs Gaussian filter for optimized settling behavior

set(filter_type: FilterTypeA) → None

```
# SCPI: [SENSe]:BWIDth:DEMod:TYPE
driver.sense.bandwidth.demod.typePy.set(filter_type = enums.FilterTypeA.FLAT)
```

This command defines the type of demodulation filter to be used. This command is identical to SENS:ADEM:BAND:DEM:TYPE:

param filter_type
FLAT Standard flat demodulation filter GAUSs Gaussian filter for optimized settling behavior

6.17.4.2 Resolution

SCPI Commands

```
SENSe:BWIDth:RESolution
```

class ResolutionCls

Resolution commands group definition. 5 total commands, 4 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:BWIDth[:RESolution]
value: float = driver.sense.bandwidth.resolution.get()
```

No command help available

return
bandwidth: No help available

set(bandwidth: float) → None

```
# SCPI: [SENSe]:BWIDth[:RESolution]
driver.sense.bandwidth.resolution.set(bandwidth = 1.0)
```

No command help available

param bandwidth
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.bandwidth.resolution.clone()
```

Subgroups

6.17.4.2.1 Auto

SCPI Commands

```
SENSe:BWIDth:RESolution:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: [SENSe]:BWIDth[:RESolution]:AUTO
driver.sense.bandwidth.resolution.auto.set(state = False)
```

No command help available

param state

No help available

6.17.4.2.2 Fft

SCPI Commands

```
SENSe:BWIDth:RESolution:FFT
```

class FftCls

Fft commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FftFilterMode

```
# SCPI: [SENSe]:BWIDth[:RESolution]:FFT
value: enums.FftFilterMode = driver.sense.bandwidth.resolution.fft.get()
```

No command help available

return

filter_mode: No help available

set(filter_mode: FftFilterMode) → None

```
# SCPI: [SENSe]:BWIDth[:RESolution]:FFT
driver.sense.bandwidth.resolution.fft.set(filter_mode = enums.FftFilterMode.
    AUTO)
```

No command help available

param filter_mode
No help available

6.17.4.2.3 Ratio

SCPI Commands

SENSe:BWIDth:RESolution:RATio

class RatioCls

Ratio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:BWIDth[:RESolution]:RATio
value: float = driver.sense.bandwidth.resolution.ratio.get()
```

No command help available

return
ratio: No help available

set(ratio: float) → None

```
# SCPI: [SENSe]:BWIDth[:RESolution]:RATio
driver.sense.bandwidth.resolution.ratio.set(ratio = 1.0)
```

No command help available

param ratio
No help available

6.17.4.2.4 TypePy

SCPI Commands

SENSe:BWIDth:RESolution:TYPE

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FilterTypeB

```
# SCPI: [SENSe]:BWIDth[:RESolution]:TYPE
value: enums.FilterTypeB = driver.sense.bandwidth.resolution.typePy.get()
```

No command help available

return
filter_type: No help available

set(filter_type: FilterTypeB) → None

```
# SCPI: [SENSe]:BWIDth[:RESolution]:TYPE
driver.sense.bandwidth.resolution.typePy.set(filter_type = enums.FilterTypeB.
↳CFILter)
```

No command help available

param filter_type
No help available

6.17.4.3 Video

SCPI Commands

```
SENSe:BWIDth:VIDeo
```

class VideoCls

Video commands group definition. 4 total commands, 3 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:BWIDth:VIDeo
value: float = driver.sense.bandwidth.video.get()
```

This command defines the video bandwidth.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic VBW selection ([SENSe:]BWIDth:VIDeo:AUTO) .

return
bandwidth: Unit: HZ

set(bandwidth: float) → None

```
# SCPI: [SENSe]:BWIDth:VIDeo
driver.sense.bandwidth.video.set(bandwidth = 1.0)
```

This command defines the video bandwidth.

INTRO_CMD_HELP: Prerequisites for this command

- Turn off automatic VBW selection ([SENSe:]BWIDth:VIDeo:AUTO) .

param bandwidth
Unit: HZ

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.bandwidth.video.clone()
```

Subgroups

6.17.4.3.1 Auto

SCPI Commands

```
SENSe:BWIDth:VIDeo:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: [SENSe]:BWIDth:VIDeo:AUTO
driver.sense.bandwidth.video.auto.set(state = False)
```

This command turns automatic selection of the video bandwidth on and off.

param state
ON | OFF | 1 | 0

6.17.4.3.2 Ratio

SCPI Commands

```
SENSe:BWIDth:VIDeo:RAtio
```

class RatioCls

Ratio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:BWIDth:VIDeo:RAtio
value: float = driver.sense.bandwidth.video.ratio.get()
```

No command help available

return
ratio: No help available

set(ratio: float) → None

```
# SCPI: [SENSe]:BWIDth:VIDeo:RAtio
driver.sense.bandwidth.video.ratio.set(ratio = 1.0)
```

No command help available

param ratio
No help available

6.17.4.3.3 TypePy

SCPI Commands

```
SENSe:BWIDth:VIDeo:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ScalingMode

```
# SCPI: [SENSe]:BWIDth:VIDeo:TYPE
value: enums.ScalingMode = driver.sense.bandwidth.video.typePy.get()
```

No command help available

return
mode: No help available

set(mode: ScalingMode) → None

```
# SCPI: [SENSe]:BWIDth:VIDeo:TYPE
driver.sense.bandwidth.video.typePy.set(mode = enums.ScalingMode.LINEar)
```

No command help available

param mode
No help available

6.17.5 Correction

SCPI Commands

```
SENSe:CORRection:RECall
```

class CorrectionCls

Correction commands group definition. 117 total commands, 6 Subgroups, 1 group commands

recall() → None

```
# SCPI: [SENSe]:CORRection:RECall
driver.sense.correction.recall()
```

No command help available

recall_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:RECall
driver.sense.correction.recall_with_opc()
```

No command help available

Same as recall, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.clone()
```

Subgroups

6.17.5.1 Collect

class CollectCls

Collect commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.collect.clone()
```

Subgroups

6.17.5.1.1 Acquire

SCPI Commands

```
SENSe:CORRection:COLlect:ACquire
```

class AcquireCls

Acquire commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(meas_type: CorrectionMeasType) → None

```
# SCPI: [SENSe]:CORRection:COLlect[:ACquire]
driver.sense.correction.collect.acquire.set(meas_type = enums.
↳ CorrectionMeasType.OPEN)
```

No command help available

param meas_type

No help available

6.17.5.2 Cvl

SCPI Commands

```
SENSe:CORRection:CVL:CLEar
```

class CvlCls

Cvl commands group definition. 11 total commands, 10 Subgroups, 1 group commands

clear() → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEar
driver.sense.correction.cvl.clear()
```

This command deletes the selected conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

clear_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:CVL:CLEar
driver.sense.correction.cvl.clear_with_opc()
```

This command deletes the selected conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

Same as clear, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.cvl.clone()
```

Subgroups

6.17.5.2.1 Band

SCPI Commands

```
SENSe:CORRection:CVL:BAND
```

class BandCls

Band commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → Band

```
# SCPI: [SENSe]:CORRection:CVL:BAND
value: enums.Band = driver.sense.correction.cvl.band.get()
```

This command defines the waveguide band for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return

band: K | KA | Q | U | V | E | W | F | D | G | Y | J | USER Standard waveguide band or user-defined band. For a definition of the frequency range for the pre-defined bands, see Table 'Frequency ranges for pre-defined bands') .

set(band: Band) → None

```
# SCPI: [SENSe]:CORRection:CVL:BAND
driver.sense.correction.cvl.band.set(band = enums.Band.A)
```

This command defines the waveguide band for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param band

K | KA | Q | U | V | E | W | F | D | G | Y | J | USER Standard waveguide band or user-defined band. For a definition of the frequency range for the pre-defined bands, see Table 'Frequency ranges for pre-defined bands') .

6.17.5.2.2 Bias

SCPI Commands

```
SENSe:CORRection:CVL:BIAS
```

class BiasCls

Bias commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
value: float = driver.sense.correction.cvl.bias.get()
```

This command defines the bias setting to be used with the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

return

bias_setting: Unit: A

set(bias_setting: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:BIAS
driver.sense.correction.cvl.bias.set(bias_setting = 1.0)
```

This command defines the bias setting to be used with the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

param bias_setting
Unit: A

6.17.5.2.3 Catalog

SCPI Commands

```
SENSe:CORRection:CVL:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:CATalog
value: float = driver.sense.correction.cvl.catalog.get()
```

This command queries all available conversion loss tables saved in the C:/R_S/INSTR/USER/cvl/ directory on the instrument. This command is only available with option B21 (External Mixer) installed.

return
catalog: 'string' Comma-separated list of strings containing the file names.

6.17.5.2.4 Comment

SCPI Commands

```
SENSe:CORRection:CVL:COMment
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:CVL:COMment
value: str = driver.sense.correction.cvl.comment.get()
```

This command defines a comment for the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return
text: No help available

set(text: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:COMment
driver.sense.correction.cvl.comment.set(text = '1')
```

This command defines a comment for the conversion loss table. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param text

No help available

6.17.5.2.5 Data

SCPI Commands

SENSe:CORRection:CVL:DATA

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Freq: List[float]: The frequencies have to be sent in ascending order. Unit: HZ
- Level: List[float]: Unit: DB

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:CVL:DATA
value: DataStruct = driver.sense.correction.cvl.data.get()
```

This command defines the reference values of the selected conversion loss tables. The values are entered as a set of frequency/level pairs. A maximum of 50 frequency/level pairs may be entered. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return

structure: for return value, see the help for DataStruct structure arguments.

set(freq: List[float], level: List[float]) → None

```
# SCPI: [SENSe]:CORRection:CVL:DATA
driver.sense.correction.cvl.data.set(freq = [1.1, 2.2, 3.3], level = [1.1, 2.2, ↵
↵3.3])
```

This command defines the reference values of the selected conversion loss tables. The values are entered as a set of frequency/level pairs. A maximum of 50 frequency/level pairs may be entered. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param freq

The frequencies have to be sent in ascending order. Unit: HZ

param level

Unit: DB

6.17.5.2.6 Harmonic

SCPI Commands

```
SENSe:CORRection:CVL:HARMonic
```

class HarmonicCls

Harmonic commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
value: float = driver.sense.correction.cvl.harmonic.get()
```

This command defines the harmonic order for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

return
harm_order: Range: 2 to 65

set(harm_order: float) → None

```
# SCPI: [SENSe]:CORRection:CVL:HARMonic
driver.sense.correction.cvl.harmonic.set(harm_order = 1.0)
```

This command defines the harmonic order for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect. This command is only available with option B21 (External Mixer) installed.

param harm_order
Range: 2 to 65

6.17.5.2.7 Mixer

SCPI Commands

```
SENSe:CORRection:CVL:MIXer
```

class MixerCls

Mixer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
value: str = driver.sense.correction.cvl.mixer.get()
```

This command defines the mixer name in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return

type_py: string Name of mixer with a maximum of 16 characters

set(type_py: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:MIXer
driver.sense.correction.cvl.mixer.set(type_py = '1')
```

This command defines the mixer name in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param type_py

string Name of mixer with a maximum of 16 characters

6.17.5.2.8 Ports

SCPI Commands

```
SENSe:CORRection:CVL:PORTs
```

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
value: int = driver.sense.correction.cvl.ports.get()
```

This command defines the mixer type in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

return

port_type: 2 | 3

set(port_type: int) → None

```
# SCPI: [SENSe]:CORRection:CVL:PORTs
driver.sense.correction.cvl.ports.set(port_type = 1)
```

This command defines the mixer type in the conversion loss table. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect) . This command is only available with option B21 (External Mixer) installed.

param port_type

2 | 3

6.17.5.2.9 Select

SCPI Commands

```
SENSe:CORRection:CVL:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filename: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:SElect
driver.sense.correction.cvl.select.set(filename = '1')
```

This command selects the conversion loss table with the specified file name. If <file_name> is not available, a new conversion loss table is created. This command is only available with option B21 (External Mixer) installed.

param filename

String containing the path and name of the file.

6.17.5.2.10 Snumber

SCPI Commands

```
SENSe:CORRection:CVL:SNUMber
```

class SnumberCls

Snumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
value: str = driver.sense.correction.cvl.snumber.get()
```

This command defines the serial number of the mixer for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect). This command is only available with option B21 (External Mixer) installed.

return

serial_no: Serial number with a maximum of 16 characters

set(serial_no: str) → None

```
# SCPI: [SENSe]:CORRection:CVL:SNUMber
driver.sense.correction.cvl.snumber.set(serial_no = '1')
```

This command defines the serial number of the mixer for which the conversion loss table is to be used. This setting is checked against the current mixer setting before the table can be assigned to the range. Before this command can be performed, the conversion loss table must be selected (see [SENSe:]CORRection:CVL:SElect). This command is only available with option B21 (External Mixer) installed.

param serial_no

Serial number with a maximum of 16 characters

6.17.5.3 Fresponse

class FresponseCls

Fresponse commands group definition. 89 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.clone()
```

Subgroups

6.17.5.3.1 Baseband

class BasebandCls

Baseband commands group definition. 24 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.clone()
```

Subgroups

6.17.5.3.1.1 User

SCPI Commands

```
SENSe:CORRection:FRESponse:BASEband:USER:PRESet
SENSe:CORRection:FRESponse:BASEband:USER:LOAD
```

class UserCls

User commands group definition. 24 total commands, 6 Subgroups, 2 group commands

load(file_path: str) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASEband:USER:LOAD
driver.sense.correction.fresponse.baseband.user.load(file_path = '1')
```

No command help available

param file_path

No help available

preset() → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:PRESet
driver.sense.correction.fresponse.baseband.user.preset()
```

No command help available

preset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:PRESet
driver.sense.correction.fresponse.baseband.user.preset_with_opc()
```

No command help available

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.clone()
```

Subgroups

6.17.5.3.1.2 Adjust

class AdjustCls

Adjust commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.adjust.clone()
```

Subgroups

6.17.5.3.1.3 RefLevel

class RefLevelCls

RefLevel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.adjust.refLevel.clone()
```

Subgroups

6.17.5.3.1.4 State

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:ADJust:RLEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:ADJust:RLEVel:STATe
value: bool = driver.sense.correction.fresponse.baseband.user.adjust.refLevel.
↳ state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:ADJust:RLEVel:STATe
driver.sense.correction.fresponse.baseband.user.adjust.refLevel.state.set(state_
↳ False)
```

No command help available

param state
No help available

6.17.5.3.1.5 Flist<FileList>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.sense.correction.fresponse.baseband.user.flist.repcap_fileList_get()
driver.sense.correction.fresponse.baseband.user.flist.repcap_fileList_set(repcap.
↳ FileList.Nr1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:CLEar
```

class FlistCls

Flist commands group definition. 8 total commands, 7 Subgroups, 1 group commands Repeated Capability: FileList, default value after init: FileList.Nr1

clear(fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:CLEar
driver.sense.correction.fresponse.baseband.user.flist.clear(fileList = repcap.
↳FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

clear_with_opc(fileList=FileList.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.flist.clone()
```

Subgroups

6.17.5.3.1.6 Catalog

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(fileList=FileList.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:CATalog
value: str = driver.sense.correction.fresponse.baseband.user.flist.catalog.
↳get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_list: No help available

6.17.5.3.1.7 Insert

SCPI Commands

`SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:INSert`

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:INSert
value: str = driver.sense.correction.fresponse.baseband.user.flist.insert.
↪get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_path: No help available

set(*file_path: str, fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:INSert
driver.sense.correction.fresponse.baseband.user.flist.insert.set(file_path = '1
↪', fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.1.8 Magnitude

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.flist.magnitude.clone()
```

Subgroups

6.17.5.3.1.9 State

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:MAGNitude:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:MAGNitude[:STATe]
value: bool = driver.sense.correction.fresponse.baseband.user.flist.magnitude.
↪ state.get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(*state: bool, fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:MAGNitude[:STATe]
driver.sense.correction.fresponse.baseband.user.flist.magnitude.state.set(state,
↪ = False, fileList = repcap.FileList.Default)
```

No command help available

param state

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.1.10 Phase

class PhaseCls

Phase commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.flist.phase.clone()
```

Subgroups

6.17.5.3.1.11 State

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:PHASe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(fileList=FileList.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:PHASe[:STATe]
value: bool = driver.sense.correction.fresponse.baseband.user.flist.phase.state.
↳get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(state: bool, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:PHASe[:STATe]
driver.sense.correction.fresponse.baseband.user.flist.phase.state.set(state =
↳False, fileList = repcap.FileList.Default)
```

No command help available

param state

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.1.12 Remove

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:REMove
```

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:REMove
driver.sense.correction.fresponse.baseband.user.flist.remove.set(fileList = ↵
↵repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

set_with_opc(*fileList=FileList.Default, opc_timeout_ms: int = -1*) → None

6.17.5.3.1.13 Select

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*file_path: str, fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:SElect
driver.sense.correction.fresponse.baseband.user.flist.select.set(file_path = '1
↵', fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.1.14 Size

SCPI Commands

`SENSe:CORRection:FRESponse:BASeband:USER:FLISt<FileList>:SIZE`

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:FLISt<fli>:SIZE
value: int = driver.sense.correction.fresponse.baseband.user.flist.size.
↪get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

size: No help available

6.17.5.3.1.15 Refresh

SCPI Commands

`SENSe:CORRection:FRESponse:BASeband:USER:REFResh`

class RefreshCls

Refresh commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:REFResh
driver.sense.correction.fresponse.baseband.user.refresh.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:REFResh
driver.sense.correction.fresponse.baseband.user.refresh.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.5.3.1.16 Slist<TouchStone>

RepCap Settings

```
# Range: Ix1 .. Ix32
rc = driver.sense.correction.fresponse.baseband.user.slist.repcap_touchStone_get()
driver.sense.correction.fresponse.baseband.user.slist.repcap_touchStone_set(repcap.
↳ TouchStone.Ix1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:CLEar
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:MOVE
```

class SlistCls

Slist commands group definition. 10 total commands, 7 Subgroups, 2 group commands Repeated Capability: TouchStone, default value after init: TouchStone.Ix1

clear(touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:CLEar
driver.sense.correction.fresponse.baseband.user.slist.clear(touchStone = repcap.
↳ TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

clear_with_opc(touchStone=TouchStone.Default, opc_timeout_ms: int = -1) → None

move(position: UpDownDirection, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:MOVE
driver.sense.correction.fresponse.baseband.user.slist.move(position = enums.
↳ UpDownDirection.DOWN, touchStone = repcap.TouchStone.Default)
```

No command help available

param position

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.slist.clone()
```

Subgroups

6.17.5.3.1.17 Catalog

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:CATalog
value: str = driver.sense.correction.fresponse.baseband.user.slist.catalog.
↳get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_list: No help available

6.17.5.3.1.18 Insert

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:INSert
```

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:INSert
value: str = driver.sense.correction.fresponse.baseband.user.slist.insert.
↳get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_path: No help available

set(file_path: str, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASEband:USER:SLISt<sli>:INSert
driver.sense.correction.fresponse.baseband.user.slist.insert.set(file_path = '1
↪', touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.1.19 Ports

class PortsCls

Ports commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.baseband.user.slist.ports.clone()
```

Subgroups

6.17.5.3.1.20 FromPy

SCPI Commands

```
SENSe:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>:PORTs:FROM
```

class FromPyCls

FromPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:BASEband:USER:SLISt<sli>:PORTs:FROM
value: int = driver.sense.correction.fresponse.baseband.user.slist.ports.fromPy.
↪get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_from: No help available

set(port_from: int, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<slI>:PORTs:FROM
driver.sense.correction.fresponse.baseband.user.slist.ports.fromPy.set(port_
↳ from = 1, touchStone = repcap.TouchStone.Default)
```

No command help available

param port_from

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.1.21 To

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:PORTs:TO
```

class ToCls

To commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<slI>:PORTs:TO
value: int = driver.sense.correction.fresponse.baseband.user.slist.ports.to.
↳ get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_to: No help available

set(port_to: int, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<slI>:PORTs:TO
driver.sense.correction.fresponse.baseband.user.slist.ports.to.set(port_to = 1,
↳ touchStone = repcap.TouchStone.Default)
```

No command help available

param port_to

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.1.22 Remove

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:REMove
```

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:REMove
driver.sense.correction.fresponse.baseband.user.slist.remove.set(touchStone = ↵
↵repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

set_with_opc(*touchStone=TouchStone.Default, opc_timeout_ms: int = -1*) → None

6.17.5.3.1.23 Select

SCPI Commands

```
SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*file_path: str, touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:SElect
driver.sense.correction.fresponse.baseband.user.slist.select.set(file_path = '1
↵', touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.1.24 Size

SCPI Commands

SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:SIZE

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:SIZE
value: int = driver.sense.correction.fresponse.baseband.user.slist.size.
↳ get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

size: No help available

6.17.5.3.1.25 State

SCPI Commands

SENSe:CORRection:FRESponse:BASeband:USER:SLISt<TouchStone>:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:STATE
value: bool = driver.sense.correction.fresponse.baseband.user.slist.state.
↳ get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

state: No help available

set(*state: bool, touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:SLISt<sli>:STATE
driver.sense.correction.fresponse.baseband.user.slist.state.set(state = False,
↳ touchStone = repcap.TouchStone.Default)
```

No command help available

param state

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.1.26 State**SCPI Commands**

SENSe:CORRection:FRESponse:BASeband:USER:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:STATe
value: bool = driver.sense.correction.fresponse.baseband.user.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:STATe
driver.sense.correction.fresponse.baseband.user.state.set(state = False)
```

No command help available

param state

No help available

6.17.5.3.1.27 Store**SCPI Commands**

SENSe:CORRection:FRESponse:BASeband:USER:STORe

class StoreCls

Store commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:BASeband:USER:STORe
driver.sense.correction.fresponse.baseband.user.store.set(file_path = '1')
```

No command help available

param file_path

No help available

6.17.5.3.2 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.correction.fresponse.inputPy.repcap_inputIx_get()
driver.sense.correction.fresponse.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 24 total commands, 1 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.clone()
```

Subgroups

6.17.5.3.2.1 User

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:PRESet
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:LOAD
```

class UserCls

User commands group definition. 24 total commands, 6 Subgroups, 2 group commands

load(file_path: str, inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:LOAD
driver.sense.correction.fresponse.inputPy.user.load(file_path = '1', inputIx = ↵
↵repcap.InputIx.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

preset(inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:PRESet
driver.sense.correction.fresponse.inputPy.user.preset(inputIx = repcap.InputIx.
↵Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

preset_with_opc(inputIx=InputIx.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.clone()
```

Subgroups**6.17.5.3.2.2 Adjust****class AdjustCls**

Adjust commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.adjust.clone()
```

Subgroups**6.17.5.3.2.3 RefLevel****class RefLevelCls**

RefLevel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.adjust.refLevel.clone()
```

Subgroups**6.17.5.3.2.4 State****SCPI Commands**

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:ADJust:RLEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*inputIx=InputIx.Default*) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:ADJust:RLEVel:STATe
value: bool = driver.sense.correction.fresponse.inputPy.user.adjust.refLevel.
↪ state.get(inputIx = repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: No help available

set(*state: bool, inputIx=InputIx.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:ADJust:RLEVel:STATe
driver.sense.correction.fresponse.inputPy.user.adjust.refLevel.state.set(state,
↪ False, inputIx = repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.17.5.3.2.5 Flist<FileList>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.sense.correction.fresponse.inputPy.user.flist.repcap_fileList_get()
driver.sense.correction.fresponse.inputPy.user.flist.repcap_fileList_set(repcap.FileList.
↪ Nr1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:CLEAr
```

class FlistCls

Flist commands group definition. 8 total commands, 7 Subgroups, 1 group commands Repeated Capability: FileList, default value after init: FileList.Nr1

clear(*inputIx=InputIx.Default, fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:CLEAr
driver.sense.correction.fresponse.inputPy.user.flist.clear(inputIx = repcap.
↪ InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

clear_with_opc(inputIx=InputIx.Default, fileList=FileList.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.flist.clone()
```

Subgroups**6.17.5.3.2.6 Catalog****SCPI Commands**

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, fileList=FileList.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:CATalog
value: str = driver.sense.correction.fresponse.inputPy.user.flist.catalog.
↳ get(inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_list: No help available

6.17.5.3.2.7 Insert

SCPI Commands

`SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:INSert`

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, fileList=FileList.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:INSert
value: str = driver.sense.correction.fresponse.inputPy.user.flist.insert.
↪ get(inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_path: No help available

set(file_path: str, inputIx=InputIx.Default, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:INSert
driver.sense.correction.fresponse.inputPy.user.flist.insert.set(file_path = '1',
↪ inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.2.8 Magnitude

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.flist.magnitude.clone()
```

Subgroups

6.17.5.3.2.9 State

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:MAGNitude:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, fileList=FileList.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:MAGNitude[:STATe]
value: bool = driver.sense.correction.fresponse.inputPy.user.flist.magnitude.
↳ state.get(inputIx = repcap.InputIx.Default, fileList = repcap.FileList.
↳ Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(state: bool, inputIx=InputIx.Default, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:MAGNitude[:STATe]
driver.sense.correction.fresponse.inputPy.user.flist.magnitude.state.set(state,
↳ False, inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.2.10 Phase**class PhaseCls**

Phase commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.flist.phase.clone()
```

Subgroups**6.17.5.3.2.11 State****SCPI Commands**

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:PHASe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, fileList=FileList.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:PHASe[:STATe]
value: bool = driver.sense.correction.fresponse.inputPy.user.flist.phase.state.
↳get(inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(state: bool, inputIx=InputIx.Default, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:PHASe[:STATe]
driver.sense.correction.fresponse.inputPy.user.flist.phase.state.set(state =
↳False, inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```


No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.2.12 Remove

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:REMove
```

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(inputIx=InputIx.Default, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:REMove
driver.sense.correction.fresponse.inputPy.user.flist.remove.set(inputIx =
↳ repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

set_with_opc(inputIx=InputIx.Default, fileList=FileList.Default, opc_timeout_ms: int = -1) → None

6.17.5.3.2.13 Select

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str, inputIx=InputIx.Default, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:SElect
driver.sense.correction.fresponse.inputPy.user.flist.select.set(file_path = '1',
↳ inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Flist’)

6.17.5.3.2.14 Size

SCPI Commands

`SENSe:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileList>:SIZE`

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, fileList=FileList.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:FLISt<fli>:SIZE
value: int = driver.sense.correction.fresponse.inputPy.user.flist.size.
↪ get(inputIx = repcap.InputIx.Default, fileList = repcap.FileList.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Flist’)

return

size: No help available

6.17.5.3.2.15 Refresh

SCPI Commands

`SENSe:CORRection:FRESponse:INPut<InputIx>:USER:REFResh`

class RefreshCls

Refresh commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:REFResh
driver.sense.correction.fresponse.inputPy.user.refresh.set(inputIx = repcap.
↪ InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

set_with_opc(inputIx=InputIx.Default, opc_timeout_ms: int = -1) → None

6.17.5.3.2.16 Slist<TouchStone>

RepCap Settings

```
# Range: Ix1 .. Ix32
rc = driver.sense.correction.fresponse.inputPy.user.slist.repcap_touchStone_get()
driver.sense.correction.fresponse.inputPy.user.slist.repcap_touchStone_set(repcap.
↪ TouchStone.Ix1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:CLEar
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:MOVE
```

class SlistCls

Slist commands group definition. 10 total commands, 7 Subgroups, 2 group commands Repeated Capability: TouchStone, default value after init: TouchStone.Ix1

clear(inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:CLEar
driver.sense.correction.fresponse.inputPy.user.slist.clear(inputIx = repcap.
↪ InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

clear_with_opc(inputIx=InputIx.Default, touchStone=TouchStone.Default, opc_timeout_ms: int = -1) → None

move(position: UpDownDirection, inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:MOVE
driver.sense.correction.fresponse.inputPy.user.slist.move(position = enums.
↪ UpDownDirection.DOWN, inputIx = repcap.InputIx.Default, touchStone = repcap.
↪ TouchStone.Default)
```

No command help available

param position

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.slist.clone()
```

Subgroups

6.17.5.3.2.17 Catalog

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:CATalog
value: str = driver.sense.correction.fresponse.inputPy.user.slist.catalog.
↳ get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_list: No help available

6.17.5.3.2.18 Insert

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:INSert
```

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:INSert
value: str = driver.sense.correction.fresponse.inputPy.user.slist.insert.
↪ get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_path: No help available

set(file_path: str, inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:INSert
driver.sense.correction.fresponse.inputPy.user.slist.insert.set(file_path = '1',
↪ inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.2.19 Ports

class PortsCls

Ports commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.inputPy.user.slist.ports.clone()
```

Subgroups

6.17.5.3.2.20 FromPy

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:PORTs:FROM
```

class FromPyCls

FromPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:PORTs:FROM
value: int = driver.sense.correction.fresponse.inputPy.user.slist.ports.fromPy.
↳get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_from: No help available

set(port_from: int, inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:PORTs:FROM
driver.sense.correction.fresponse.inputPy.user.slist.ports.fromPy.set(port_from,
↳= 1, inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param port_from

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.2.21 To**SCPI Commands**

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:PORTs:TO
```

class ToCls

To commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:PORTs:TO
value: int = driver.sense.correction.fresponse.inputPy.user.slist.ports.to.
↳get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_to: No help available

set(port_to: int, inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:PORTs:TO
driver.sense.correction.fresponse.inputPy.user.slist.ports.to.set(port_to = 1,
↳inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param port_to

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.2.22 Remove

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:REMove
```

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*inputIx=InputIx.Default, touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:REMove
driver.sense.correction.fresponse.inputPy.user.slist.remove.set(inputIx = ↵
↵ repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

set_with_opc(*inputIx=InputIx.Default, touchStone=TouchStone.Default, opc_timeout_ms: int = -1*) → None

6.17.5.3.2.23 Select

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*file_path: str, inputIx=InputIx.Default, touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:SElect
driver.sense.correction.fresponse.inputPy.user.slist.select.set(file_path = '1',
↵ inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.2.24 Size

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:SIZE
value: int = driver.sense.correction.fresponse.inputPy.user.slist.size.
↪get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

size: No help available

6.17.5.3.2.25 State

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<TouchStone>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default, touchStone=TouchStone.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:STATe
value: bool = driver.sense.correction.fresponse.inputPy.user.slist.state.
↪get(inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

state: No help available

set(state: bool, inputIx=InputIx.Default, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:SLISt<sli>:STATe
driver.sense.correction.fresponse.inputPy.user.slist.state.set(state = False, ↵
↵ inputIx = repcap.InputIx.Default, touchStone = repcap.TouchStone.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Slist’)

6.17.5.3.2.26 State

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:STATe
value: bool = driver.sense.correction.fresponse.inputPy.user.state.get(inputIx, ↵
↵ repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:STATe
driver.sense.correction.fresponse.inputPy.user.state.set(state = False, inputIx, ↵
↵ repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘InputPy’)

6.17.5.3.2.27 Store

SCPI Commands

```
SENSe:CORRection:FRESponse:INPut<InputIx>:USER:STORe
```

class StoreCls

Store commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str, inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:INPut<ip>:USER:STORe
driver.sense.correction.fresponse.inputPy.user.store.set(file_path = '1',
↪inputIx = repcap.InputIx.Default)
```

No command help available

param file_path

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.17.5.3.3 Lsources

class LsourcesCls

Lsources commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.lsources.clone()
```

Subgroups

6.17.5.3.3.1 State

SCPI Commands

```
SENSe:CORRection:FRESponse:LSources:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:LSources:STATe
value: bool = driver.sense.correction.fresponse.lsources.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:LSources:STATE
driver.sense.correction.fresponse.lsources.state.set(state = False)
```

No command help available

param state

No help available

6.17.5.3.4 User

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:PRESet
SENSe:CORRection:FRESponse:USER:LOAD
```

class UserCls

User commands group definition. 40 total commands, 12 Subgroups, 2 group commands

load(file_path: str) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:LOAD
driver.sense.correction.fresponse.user.load(file_path = '1')
```

No command help available

param file_path

No help available

preset() → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:PRESet
driver.sense.correction.fresponse.user.preset()
```

No command help available

preset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:PRESet
driver.sense.correction.fresponse.user.preset_with_opc()
```

No command help available

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.clone()
```

Subgroups

6.17.5.3.4.1 Adjust

class AdjustCls

Adjust commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.adjust.clone()
```

Subgroups

6.17.5.3.4.2 RefLevel

class RefLevelCls

RefLevel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.adjust.refLevel.clone()
```

Subgroups

6.17.5.3.4.3 State

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:ADJust:RLEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:ADJust:RLEVel:STATe
value: bool = driver.sense.correction.fresponse.user.adjust.refLevel.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:ADJust:RLEVel:STATe
driver.sense.correction.fresponse.user.adjust.refLevel.state.set(state = False)
```

No command help available

param state

No help available

6.17.5.3.4.4 Flist<FileList>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.sense.correction.fresponse.user.flist.repcap_fileList_get()
driver.sense.correction.fresponse.user.flist.repcap_fileList_set(repcap.FileList.Nr1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:CLEar
```

class FlistCls

Flist commands group definition. 11 total commands, 8 Subgroups, 1 group commands Repeated Capability: FileList, default value after init: FileList.Nr1

clear(fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:CLEar
driver.sense.correction.fresponse.user.flist.clear(fileList = repcap.FileList.
↳Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

clear_with_opc(fileList=FileList.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.flist.clone()
```

Subgroups

6.17.5.3.4.5 Catalog

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(fileList=FileList.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:CATalog
value: str = driver.sense.correction.fresponse.user.flist.catalog.get(fileList,
↳= repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_list: No help available

6.17.5.3.4.6 Data

class DataCls

Data commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.flist.data.clone()
```

Subgroups

6.17.5.3.4.7 Frequency

SCPI Commands

`SENSe:CORRection:FRESponse:USER:FLISt<FileList>:DATA:FREQuency`

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:DATA:FREQuency
value: float = driver.sense.correction.fresponse.user.flist.data.frequency.
↪get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

result: No help available

6.17.5.3.4.8 Magnitude

SCPI Commands

`SENSe:CORRection:FRESponse:USER:FLISt<FileList>:DATA:MAGNitude`

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:DATA:MAGNitude
value: float = driver.sense.correction.fresponse.user.flist.data.magnitude.
↪get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

result: No help available

6.17.5.3.4.9 Phase

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:DATA:PHASe
```

class PhaseCls

Phase commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:DATA:PHASe
value: float = driver.sense.correction.fresponse.user.flist.data.phase.
↳get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

result: No help available

6.17.5.3.4.10 Insert

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:INSert
```

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:INSert
value: str = driver.sense.correction.fresponse.user.flist.insert.get(fileList =
↳repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

file_path: No help available

set(*file_path: str, fileList=FileList.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:INSert
driver.sense.correction.fresponse.user.flist.insert.set(file_path = '1',
↳fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.4.11 Magnitude

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.flist.magnitude.clone()
```

Subgroups

6.17.5.3.4.12 State

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:MAGNitude:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(fileList=FileList.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:MAGNitude[:STATe]
value: bool = driver.sense.correction.fresponse.user.flist.magnitude.state.
↳get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(state: bool, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:MAGNitude[:STATe]
driver.sense.correction.fresponse.user.flist.magnitude.state.set(state = False,
↳fileList = repcap.FileList.Default)
```

No command help available

param state

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.4.13 Phase**class PhaseCls**

Phase commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.flist.phase.clone()
```

Subgroups**6.17.5.3.4.14 State****SCPI Commands**

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:PHASe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(fileList=FileList.Default) → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:PHASe[:STATe]
value: bool = driver.sense.correction.fresponse.user.flist.phase.state.
↳ get(fileList = repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

state: No help available

set(state: bool, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:PHASe[:STATe]
driver.sense.correction.fresponse.user.flist.phase.state.set(state = False,
↳ fileList = repcap.FileList.Default)
```

No command help available

param state

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.4.15 Remove**SCPI Commands**

SENSe:CORRection:FRESponse:USER:FLISt<FileList>:REMove

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:REMove
driver.sense.correction.fresponse.user.flist.remove.set(fileList = repcap.
↳FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

set_with_opc(fileList=FileList.Default, opc_timeout_ms: int = -1) → None

6.17.5.3.4.16 Select**SCPI Commands**

SENSe:CORRection:FRESponse:USER:FLISt<FileList>:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str, fileList=FileList.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:SElect
driver.sense.correction.fresponse.user.flist.select.set(file_path = '1',
↳fileList = repcap.FileList.Default)
```

No command help available

param file_path

No help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

6.17.5.3.4.17 Size

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FLISt<FileList>:SIZE
```

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*fileList=FileList.Default*) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FLISt<fli>:SIZE
value: int = driver.sense.correction.fresponse.user.flist.size.get(fileList = ↵
↵repcap.FileList.Default)
```

No command help available

param fileList

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Flist')

return

size: No help available

6.17.5.3.4.18 Fstate

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:FSTate
```

class FstateCls

Fstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FSTate
value: bool = driver.sense.correction.fresponse.user.fstate.get()
```

No command help available

return

state: No help available

set(*state: bool*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:FSTate
driver.sense.correction.fresponse.user.fstate.set(state = False)
```

No command help available

param state

No help available

6.17.5.3.4.19 Iq

class IqCls

Iq commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.iq.clone()
```

Subgroups

6.17.5.3.4.20 Data

class DataCls

Data commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.iq.data.clone()
```

Subgroups

6.17.5.3.4.21 Frequency

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:IQ:DATA:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:IQ:DATA:FREQuency
value: List[float] = driver.sense.correction.fresponse.user.iq.data.frequency.
    ↪get()
```

No command help available

```
    return
        trace_data: No help available
```

6.17.5.3.4.22 Magnitude

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:IQ:DATA:MAGNitude
```

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:IQ:DATA:MAGNitude
value: List[float] = driver.sense.correction.fresponse.user.iq.data.magnitude.
↳get()
```

No command help available

```
return
    trace_data: No help available
```

6.17.5.3.4.23 Phase

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:IQ:DATA:PHASe
```

class PhaseCls

Phase commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:IQ:DATA:PHASe
value: List[float] = driver.sense.correction.fresponse.user.iq.data.phase.get()
```

No command help available

```
return
    trace_data: No help available
```

6.17.5.3.4.24 Pstate

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:PState
```

class PstateCls

Pstate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:PState
value: bool = driver.sense.correction.fresponse.user.pstate.get()
```

No command help available

return

state: No help available

set(*state: bool*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:PState
driver.sense.correction.fresponse.user.pstate.set(state = False)
```

No command help available

param state

No help available

6.17.5.3.4.25 Scope

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SCOpe
```

class ScopeCls

Scope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FramesScope

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SCOpe
value: enums.FramesScope = driver.sense.correction.fresponse.user.scope.get()
```

No command help available

return

scope: No help available

set(*scope: FramesScope*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SCOpe
driver.sense.correction.fresponse.user.scope.set(scope = enums.FramesScope.ALL)
```

No command help available

param scope

No help available

6.17.5.3.4.26 Scovered

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SCOVered
```

class ScoveredCls

Scovered commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SCOVeRed
value: float = driver.sense.correction.fresponse.user.scovered.get()
```

No command help available

return
covered: No help available

6.17.5.3.4.27 Slist<TouchStone>

RepCap Settings

```
# Range: Ix1 .. Ix32
rc = driver.sense.correction.fresponse.user.slist.repcap_touchStone_get()
driver.sense.correction.fresponse.user.slist.repcap_touchStone_set(repcap.TouchStone.Ix1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:CLEar
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:MOVE
```

class SlistCls

Slist commands group definition. 13 total commands, 8 Subgroups, 2 group commands Repeated Capability: TouchStone, default value after init: TouchStone.Ix1

clear(touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:CLEar
driver.sense.correction.fresponse.user.slist.clear(touchStone = repcap.
↳ TouchStone.Default)
```

No command help available

param touchStone
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

clear_with_opc(touchStone=TouchStone.Default, opc_timeout_ms: int = -1) → None

move(position: UpDownDirection, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:MOVE
driver.sense.correction.fresponse.user.slist.move(position = enums.
↳ UpDownDirection.DOWN, touchStone = repcap.TouchStone.Default)
```

No command help available

param position
No help available

param touchStone
optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.slist.clone()
```

Subgroups

6.17.5.3.4.28 Catalog

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:CATalog
value: str = driver.sense.correction.fresponse.user.slist.catalog.
↳get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_list: No help available

6.17.5.3.4.29 Data

class DataCls

Data commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.slist.data.clone()
```

Subgroups

6.17.5.3.4.30 Frequency

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:DATA:FREQuency<SPortPair>
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone*=*TouchStone.Default*, *sPortPair*=*SPortPair.Ix1*) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:DATA:FREQuency<spi>
value: float = driver.sense.correction.fresponse.user.slist.data.frequency.
↳get(touchStone = repcap.TouchStone.Default, sPortPair = repcap.SPortPair.Ix1)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

param sPortPair

optional repeated capability selector. Default value: Ix1

return

result: No help available

6.17.5.3.4.31 Magnitude

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:DATA:MAGNitude<SPortPair>
```

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone*=*TouchStone.Default*, *sPortPair*=*SPortPair.Ix1*) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:DATA:MAGNitude<spi>
value: float = driver.sense.correction.fresponse.user.slist.data.magnitude.
↳get(touchStone = repcap.TouchStone.Default, sPortPair = repcap.SPortPair.Ix1)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

param sPortPair

optional repeated capability selector. Default value: Ix1

return

result: No help available

6.17.5.3.4.32 Phase<SPortPair>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.sense.correction.fresponse.user.slist.data.phase.repcap_sPortPair_get()
driver.sense.correction.fresponse.user.slist.data.phase.repcap_sPortPair_set(repcap.
↳ SPortPair.Ix1)
```

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:DATA:PHASe<SPortPair>
```

class PhaseCls

Phase commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SPortPair, default value after init: SPortPair.Ix1

get(touchStone=TouchStone.Default, sPortPair=SPortPair.Default) → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:DATA:PHASe<spi>
value: float = driver.sense.correction.fresponse.user.slist.data.phase.
↳ get(touchStone = repcap.TouchStone.Default, sPortPair = repcap.SPortPair.
↳ Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

param sPortPair

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Phase')

return

result: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.slist.data.phase.clone()
```

6.17.5.3.4.33 Insert

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:INSert
```

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(touchStone=TouchStone.Default) → str

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:INSert
value: str = driver.sense.correction.fresponse.user.slist.insert.get(touchStone,
↪= repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

file_path: No help available

set(file_path: str, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:INSert
driver.sense.correction.fresponse.user.slist.insert.set(file_path = '1',
↪touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.4.34 Ports

class PortsCls

Ports commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.slist.ports.clone()
```

Subgroups

6.17.5.3.4.35 FromPy

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:PORTs:FROM
```

class FromPyCls

FromPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:PORTs:FROM
value: int = driver.sense.correction.fresponse.user.slist.ports.fromPy.
↪get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_from: No help available

set(*port_from: int, touchStone=TouchStone.Default*) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:PORTs:FROM
driver.sense.correction.fresponse.user.slist.ports.fromPy.set(port_from = 1,↪
↪touchStone = repcap.TouchStone.Default)
```

No command help available

param port_from

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.4.36 To

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:PORTs:TO
```

class ToCls

To commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → int

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:PORTs:TO
value: int = driver.sense.correction.fresponse.user.slist.ports.to.
↪get(touchStone = repcap.TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

port_to: No help available

set(port_to: int, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:PORTs:TO
driver.sense.correction.fresponse.user.slist.ports.to.set(port_to = 1,↵
↵touchStone = repcap.TouchStone.Default)
```

No command help available

param port_to

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.4.37 Remove

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:REMove
```

class RemoveCls

Remove commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:REMove
driver.sense.correction.fresponse.user.slist.remove.set(touchStone = repcap.
↵TouchStone.Default)
```

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

set_with_opc(touchStone=TouchStone.Default, opc_timeout_ms: int = -1) → None

6.17.5.3.4.38 Select

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:SElect
driver.sense.correction.fresponse.user.slist.select.set(file_path = '1',↵
↵touchStone = repcap.TouchStone.Default)
```

No command help available

param file_path

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.4.39 Size**SCPI Commands**

SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:SIZE

class SizeCls

Size commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → int

SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:SIZE
value: int = driver.sense.correction.fresponse.user.slist.size.get(touchStone =
↪repcap.TouchStone.Default)

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

size: No help available

6.17.5.3.4.40 State**SCPI Commands**

SENSe:CORRection:FRESponse:USER:SLISt<TouchStone>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*touchStone=TouchStone.Default*) → bool

SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:STATe
value: bool = driver.sense.correction.fresponse.user.slist.state.get(touchStone↪
↪= repcap.TouchStone.Default)

No command help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

return

state: No help available

set(state: bool, touchStone=TouchStone.Default) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SLISt<slI>:STATE
driver.sense.correction.fresponse.user.slist.state.set(state = False,
↪ touchStone = repcap.TouchStone.Default)
```

No command help available

param state

No help available

param touchStone

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Slist')

6.17.5.3.4.41 Spectrum

class SpectrumCls

Spectrum commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.spectrum.clone()
```

Subgroups

6.17.5.3.4.42 Data

class DataCls

Data commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.fresponse.user.spectrum.data.clone()
```

Subgroups

6.17.5.3.4.43 Frequency

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:SPECTrum:DATA:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SPECTrum:DATA:FREQuency
value: List[float] = driver.sense.correction.fresponse.user.spectrum.data.
↪ frequency.get()
```

No command help available

```
return
    trace_data: No help available
```

6.17.5.3.4.44 Magnitude

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:SPECTrum:DATA:MAGNitude
```

class MagnitudeCls

Magnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SPECTrum:DATA:MAGNitude
value: List[float] = driver.sense.correction.fresponse.user.spectrum.data.
↪ magnitude.get()
```

No command help available

```
return
    trace_data: No help available
```

6.17.5.3.4.45 Phase

SCPI Commands

```
FORMAT REAL, 32;SENSe:CORRection:FRESponse:USER:SPECTrum:DATA:PHASe
```

class PhaseCls

Phase commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:SPECTrum:DATA:PHASe
value: List[float] = driver.sense.correction.fresponse.user.spectrum.data.phase.
↪ get()
```

No command help available

```
return
    trace_data: No help available
```

6.17.5.3.4.46 State

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:STATe
value: bool = driver.sense.correction.fresponse.user.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:STATe
driver.sense.correction.fresponse.user.state.set(state = False)
```

No command help available

param state
No help available

6.17.5.3.4.47 Store

SCPI Commands

```
SENSe:CORRection:FRESponse:USER:STORe
```

class StoreCls

Store commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(file_path: str) → None

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:STORe
driver.sense.correction.fresponse.user.store.set(file_path = '1')
```

No command help available

param file_path
No help available

6.17.5.3.4.48 Valid

SCPI Commands

SENSe:CORRection:FRESponse:USER:VALid

class ValidCls

Valid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:CORRection:FRESponse:USER:VALid
value: float = driver.sense.correction.fresponse.user.valid.get()
```

No command help available

return
validity: No help available

6.17.5.4 Method

SCPI Commands

SENSe:CORRection:METHod

class MethodCls

Method commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CorrectionMethod

```
# SCPI: [SENSe]:CORRection:METHod
value: enums.CorrectionMethod = driver.sense.correction.method.get()
```

No command help available

return
type_py: No help available

set(type_py: *CorrectionMethod*) → None

```
# SCPI: [SENSe]:CORRection:METHod
driver.sense.correction.method.set(type_py = enums.CorrectionMethod.REFlexion)
```

No command help available

param type_py
No help available

6.17.5.5 State

SCPI Commands

```
SENSe:CORRection:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection[:STATe]
value: bool = driver.sense.correction.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection[:STATe]
driver.sense.correction.state.set(state = False)
```

No command help available

param state
No help available

6.17.5.6 Transducer

SCPI Commands

```
SENSe:CORRection:TRANsducer:DELeTe
```

class TransducerCls

Transducer commands group definition. 13 total commands, 11 Subgroups, 1 group commands

delete() → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:DELeTe
driver.sense.correction.transducer.delete()
```

No command help available

delete_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:DELeTe
driver.sense.correction.transducer.delete_with_opc()
```

No command help available

Same as delete, but waits for the operation to complete before continuing further. Use the RsF-swp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.transducer.clone()
```

Subgroups

6.17.5.6.1 Active

SCPI Commands

```
SENSe:CORRection:TRANsducer:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:TRANsducer:ACTive
value: str = driver.sense.correction.transducer.active.get()
```

No command help available

```
return
    transducer_factor: No help available
```

6.17.5.6.2 Adjust

class AdjustCls

Adjust commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.transducer.adjust.clone()
```

Subgroups

6.17.5.6.2.1 RefLevel

class RefLevelCls

RefLevel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.transducer.adjust.refLevel.clone()
```

Subgroups

6.17.5.6.2.2 State

SCPI Commands

```
SENSe:CORRection:TRANsducer:ADJust:RLEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:TRANsducer:ADJust:RLEVel[:STATe]
value: bool = driver.sense.correction.transducer.adjust.refLevel.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:ADJust:RLEVel[:STATe]
driver.sense.correction.transducer.adjust.refLevel.state.set(state = False)
```

No command help available

param state

No help available

6.17.5.6.3 Catalog

SCPI Commands

```
SENSe:CORRection:TRANsducer:CATalog
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Used_Disk_Space: int: No parameter help available
- Free_Disk_Space: int: No parameter help available
- File_Size: List[int]: No parameter help available

- Filename: List[str]: No parameter help available

get() → GetStruct

```
# SCPI: [SENSe]:CORRection:TRANsducer:CATalog
value: GetStruct = driver.sense.correction.transducer.catalog.get()
```

No command help available

return

structure: for return value, see the help for GetStruct structure arguments.

6.17.5.6.4 Comment

SCPI Commands

```
SENSe:CORRection:TRANsducer:COMMeNt
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:TRANsducer:COMMeNt
value: str = driver.sense.correction.transducer.comment.get()
```

No command help available

return

comment: No help available

set(comment: str) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:COMMeNt
driver.sense.correction.transducer.comment.set(comment = '1')
```

No command help available

param comment

No help available

6.17.5.6.5 Data

SCPI Commands

```
SENSe:CORRection:TRANsducer:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DataStruct

Response structure. Fields:

- Frequency: List[float]: No parameter help available

- Level: List[float]: No parameter help available

get() → DataStruct

```
# SCPI: [SENSe]:CORRection:TRANsducer:DATA
value: DataStruct = driver.sense.correction.transducer.data.get()
```

No command help available

return

structure: for return value, see the help for DataStruct structure arguments.

set(frequency: List[float], level: List[float]) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:DATA
driver.sense.correction.transducer.data.set(frequency = [1.1, 2.2, 3.3], level_
↪= [1.1, 2.2, 3.3])
```

No command help available

param frequency

No help available

param level

No help available

6.17.5.6.6 Generate

SCPI Commands

```
SENSe:CORRection:TRANsducer:GENerate
```

class GenerateCls

Generate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:TRANsducer:GENerate
value: str = driver.sense.correction.transducer.generate.get()
```

No command help available

return

name: No help available

set(name: str) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:GENerate
driver.sense.correction.transducer.generate.set(name = '1')
```

No command help available

param name

No help available

6.17.5.6.7 InputPy<InputIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.correction.transducer.inputPy.repcap_inputIx_get()
driver.sense.correction.transducer.inputPy.repcap_inputIx_set(repcap.InputIx.Nr1)
```

class InputPyCls

InputPy commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: InputIx, default value after init: InputIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.correction.transducer.inputPy.clone()
```

Subgroups

6.17.5.6.7.1 Active

SCPI Commands

```
SENSe:CORRection:TRANsducer:INPut<InputIx>:ACTive
```

class ActiveCls

Active commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → str

```
# SCPI: [SENSe]:CORRection:TRANsducer:INPut<ip>:ACTive
value: str = driver.sense.correction.transducer.inputPy.active.get(inputIx =
↳ repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

transducer_factor: No help available

6.17.5.6.7.2 State

SCPI Commands

```
SENSe:CORRection:TRANsdUCer:INPut<InputIx>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(inputIx=InputIx.Default) → bool

```
# SCPI: [SENSe]:CORRection:TRANsdUCer:INPut<ip>[:STATe]
value: bool = driver.sense.correction.transducer.inputPy.state.get(inputIx =
↳repcap.InputIx.Default)
```

No command help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

return

state: No help available

set(state: bool, inputIx=InputIx.Default) → None

```
# SCPI: [SENSe]:CORRection:TRANsdUCer:INPut<ip>[:STATe]
driver.sense.correction.transducer.inputPy.state.set(state = False, inputIx =
↳repcap.InputIx.Default)
```

No command help available

param state

No help available

param inputIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'InputPy')

6.17.5.6.8 Scaling

SCPI Commands

```
SENSe:CORRection:TRANsdUCer:SCALing
```

class ScalingCls

Scaling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ScalingMode

```
# SCPI: [SENSe]:CORRection:TRANsdUCer:SCALing
value: enums.ScalingMode = driver.sense.correction.transducer.scaling.get()
```

No command help available

return

scaling_type: No help available

set(scaling_type: ScalingMode) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:SCALing
driver.sense.correction.transducer.scaling.set(scaling_type = enums.ScalingMode.
↳LINear)
```

No command help available

param scaling_type

No help available

6.17.5.6.9 Select

SCPI Commands

```
SENSe:CORRection:TRANsducer:SElect
```

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(name: str) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:SElect
driver.sense.correction.transducer.select.set(name = '1')
```

No command help available

param name

No help available

6.17.5.6.10 State

SCPI Commands

```
SENSe:CORRection:TRANsducer:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:CORRection:TRANsducer[:STATe]
value: bool = driver.sense.correction.transducer.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer[:STATe]
driver.sense.correction.transducer.state.set(state = False)
```

No command help available

param state

No help available

6.17.5.6.11 Unit

SCPI Commands

```
SENSe:CORRection:TRANsducer:UNIT
```

class UnitCls

Unit commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:CORRection:TRANsducer:UNIT
value: str = driver.sense.correction.transducer.unit.get()
```

No command help available

return

unit: No help available

set(unit: str) → None

```
# SCPI: [SENSe]:CORRection:TRANsducer:UNIT
driver.sense.correction.transducer.unit.set(unit = '1')
```

No command help available

param unit

No help available

6.17.6 Ddemod

class DdemodCls

Ddemod commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ddemod.clone()
```

Subgroups

6.17.6.1 Search

class SearchCls

Search commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ddemod.search.clone()
```

Subgroups

6.17.6.1.1 Sync

class SyncCls

Sync commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.ddemod.search.sync.clone()
```

Subgroups

6.17.6.1.1.1 State

SCPI Commands

```
SENSe:DDEMod:SEARch:SYNC:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:DDEMod:SEARch:SYNC[:STATe]
value: bool = driver.sense.ddemod.search.sync.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:DDEMod:SEARCh:SYNC[:STATe]
driver.sense.ddemod.search.sync.state.set(state = False)
```

No command help available

param state

No help available

6.17.7 Espectrum<SubBlock>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.sense.espectrum.repcap_subBlock_get()
driver.sense.espectrum.repcap_subBlock_set(repcap.SubBlock.Nr1)
```

class EspectrumCls

Espectrum commands group definition. 47 total commands, 11 Subgroups, 0 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.clone()
```

Subgroups

6.17.7.1 Bwid

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:BWID
```

class BwidCls

Bwid commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:BWID
value: float = driver.sense.espectrum.bwid.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

bandwidth: No help available

set(bandwidth: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:BWID
driver.sense.espectrum.bwid.set(bandwidth = 1.0, subBlock = repcap.SubBlock.
↪Default)
```

No command help available

param bandwidth

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.2 FilterPy

class FilterPyCls

FilterPy commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.filterPy.clone()
```

Subgroups

6.17.7.2.1 Rrc

class RrcCls

Rrc commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.filterPy.rrc.clone()
```

Subgroups

6.17.7.2.1.1 Alpha

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:FILTer:RRC:ALPHA
```


class AlphaCls

Alpha commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:FILTer[:RRC]:ALPHa
value: float = driver.sense.espectrum.filterPy.rrc.alpha.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

alpha: No help available

set(alpha: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:FILTer[:RRC]:ALPHa
driver.sense.espectrum.filterPy.rrc.alpha.set(alpha = 1.0, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param alpha

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.2.1.2 State**SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:FILTer:RRC:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:FILTer[:RRC][:STATe]
value: bool = driver.sense.espectrum.filterPy.rrc.state.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:FILTer[:RRC][:STATe]
driver.sense.spectrum.filterPy.rrc.state.set(state = False, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.3 Hspeed

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:HSPEED
```

class HspeedCls

Hspeed commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:HSPEED
value: bool = driver.sense.spectrum.hspeed.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:HSPEED
driver.sense.spectrum.hspeed.set(state = False, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4 Msr

class MsrCls

Msr commands group definition. 9 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.msr.clone()
```

Subgroups

6.17.7.4.1 Apply

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:MSR:APPLY
```

class ApplyCls

Apply commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:APPLY
driver.sense.espectrum.msr.apply.set(subBlock = repcap.SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

set_with_opc(subBlock=SubBlock.Default, opc_timeout_ms: int = -1) → None

6.17.7.4.2 Band

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:MSR:BAND
```

class BandCls

Band commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → LowHigh

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:BAND
value: enums.LowHigh = driver.sense.espectrum.msr.band.get(subBlock = repcap.
↳ SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

range_py: No help available

set(range_py: LowHigh, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:BAND
driver.sense.espectrum.msr.band.set(range_py = enums.LowHigh.HIGH, subBlock =
↪repcap.SubBlock.Default)
```

No command help available

param range_py

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4.3 Bcategory

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:MSR:BCATegory
```

class BcategoryCls

Bcategory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:BCATegory
value: float = driver.sense.espectrum.msr.bcategory.get(subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

category: No help available

set(category: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:BCATegory
driver.sense.espectrum.msr.bcategory.set(category = 1.0, subBlock = repcap.
↪SubBlock.Default)
```

No command help available

param category

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4.4 ClassPy**SCPI Commands**

SENSe:ESpectrum<SubBlock>:MSR:CLASs

class ClassPyCls

ClassPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → RangeClass

<pre># SCPI: [SENSe]:ESpectrum<sb>:MSR:CLASs value: enums.RangeClass = driver.sense.espectrum.msr.classPy.get(subBlock = ↵ ↵repcap.SubBlock.Default)</pre>
--

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

class_py: No help available

set(class_py: RangeClass, subBlock=SubBlock.Default) → None

<pre># SCPI: [SENSe]:ESpectrum<sb>:MSR:CLASs driver.sense.espectrum.msr.classPy.set(class_py = enums.RangeClass.L0Cal, ↵ ↵subBlock = repcap.SubBlock.Default)</pre>

No command help available

param class_py

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4.5 Gsm**class GsmCls**

Gsm commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.msr.gsm.clone()
```

Subgroups

6.17.7.4.5.1 Carrier

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:MSR:GSM:CARRier
```

class CarrierCls

Carrier commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:GSM:CARRier
value: float = driver.sense.espectrum.msr.gsm.carrier.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

power: No help available

set(power: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:GSM:CARRier
driver.sense.espectrum.msr.gsm.carrier.set(power = 1.0, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param power

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4.5.2 Cpresent

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:MSR:GSM:CPresent
```

class CpresentCls

Cpresent commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:GSM:CPresent
value: bool = driver.sense.espectrum.msr.gsm.cpresent.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:GSM:CPresent
driver.sense.espectrum.msr.gsm.cpresent.set(state = False, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.4.6 Lte

class LteCls

Lte commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.msr.lte.clone()
```

Subgroups

6.17.7.4.6.1 Cpresent

SCPI Commands

SENSe:ESpectrum<SubBlock>:MSR:LTE:CPresent

class CpresentCls

Cpresent commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:LTE:CPresent
value: bool = driver.sense.espectrum.msr.lte.cpresent.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return

state: No help available

set(state: bool, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:LTE:CPresent
driver.sense.espectrum.msr.lte.cpresent.set(state = False, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

6.17.7.4.7 Mpower

SCPI Commands

SENSe:ESpectrum<SubBlock>:MSR:MPower

class MpowerCls

Mpower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:MPower
value: float = driver.sense.espectrum.msr.mpower.get(subBlock = repcap.SubBlock.
↳Default)
```


No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return

power: No help available

set(*power: float, subBlock=SubBlock.Default*) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:MPower
driver.sense.espectrum.msr.mpower.set(power = 1.0, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param power

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

6.17.7.4.8 RfbWidth

SCPI Commands

SENSe:ESpectrum<SubBlock>:MSR:RFBWidth

class RfbWidthCls

RfbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*subBlock=SubBlock.Default*) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:RFBWidth
value: float = driver.sense.espectrum.msr.rfbWidth.get(subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return

bandwidth: No help available

set(*bandwidth: float, subBlock=SubBlock.Default*) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:MSR:RFBWidth
driver.sense.espectrum.msr.rfbWidth.set(bandwidth = 1.0, subBlock = repcap.
↳SubBlock.Default)
```

No command help available

param bandwidth

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.5 Preset

class PresetCls

Preset commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.preset.clone()
```

Subgroups

6.17.7.5.1 Restore

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:PRESet:REStore
```

class RestoreCls

Restore commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:PRESet:REStore
driver.sense.espectrum.preset.restore.set(subBlock = repcap.SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

set_with_opc(subBlock=SubBlock.Default, opc_timeout_ms: int = -1) → None

6.17.7.5.2 Standard

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:PRESet:STANdard
```

class StandardCls

Standard commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → str

```
# SCPI: [SENSe]:ESpectrum<sb>:PRESet[:STANdard]
value: str = driver.sense.espectrum.preset.standard.get(subBlock = repcap.
↳ SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

standard: No help available

set(standard: str, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:PRESet[:STANdard]
driver.sense.espectrum.preset.standard.set(standard = '1', subBlock = repcap.
↳ SubBlock.Default)
```

No command help available

param standard

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.5.3 Store

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:PRESet:STORe
```

class StoreCls

Store commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → str

```
# SCPI: [SENSe]:ESpectrum<sb>:PRESet:STORe
value: str = driver.sense.espectrum.preset.store.get(subBlock = repcap.SubBlock.
↳ Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

standard: No help available

set(*standard: str, subBlock=SubBlock.Default*) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:PRESet:STORe
driver.sense.espectrum.preset.store.set(standard = '1', subBlock = repcap.
↳ SubBlock.Default)
```

No command help available

param standard

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.6 Range<RangePy>

RepCap Settings

```
# Range: Ix1 .. Ix64
rc = driver.sense.espectrum.range.repcap_rangePy_get()
driver.sense.espectrum.range.repcap_rangePy_set(repcap.RangePy.Ix1)
```

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:DELeTe
```

class RangeCls

Range commands group definition. 26 total commands, 11 Subgroups, 1 group commands Repeated Capability: RangePy, default value after init: RangePy.Ix1

delete(*subBlock=SubBlock.Default, rangePy=RangePy.Default*) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:DELeTe
driver.sense.espectrum.range.delete(subBlock = repcap.SubBlock.Default, rangePy.
↳ repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

delete_with_opc(*subBlock=SubBlock.Default, rangePy=RangePy.Default, opc_timeout_ms: int = -1*) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.clone()
```

Subgroups

6.17.7.6.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.bandwidth.clone()
```

Subgroups

6.17.7.6.1.1 Resolution

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:BANdwidth:RESolution
```

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:BANdwidth:RESolution
value: float = driver.sense.spectrum.range.bandwidth.resolution.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

rbw: No help available

set(rbw: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:BANdwidth:RESolution
driver.sense.spectrum.range.bandwidth.resolution.set(rbw = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param rbw

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.1.2 Video

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:BANdwidth:VIDeo
```

class VideoCls

Video commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:BANdwidth:VIDeo
value: float = driver.sense.spectrum.range.bandwidth.video.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

vbw: No help available

set(vbw: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:BANdwidth:VIDeo
driver.sense.spectrum.range.bandwidth.video.set(vbw = 1.0, subBlock = repcap.
↳SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param vbw

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.2 Count**SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:COUNT
value: float = driver.sense.espectrum.range.count.get(subBlock = repcap.
↳ SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

ranges: No help available

6.17.7.6.3 FilterPy**class FilterPyCls**

FilterPy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.range.filterPy.clone()
```

Subgroups

6.17.7.6.3.1 TypePy

SCPI Commands

`SENSe:ESpectrum<SubBlock>:RANge<RangePy>:FILTer:TYPE`

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → FilterTypeC

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:FILTer:TYPE
value: enums.FilterTypeC = driver.sense.espectrum.range.filterPy.typePy.
↳get(subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

filter_type: No help available

set(filter_type: FilterTypeC, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:FILTer:TYPE
driver.sense.espectrum.range.filterPy.typePy.set(filter_type = enums.
↳FilterTypeC.CFilter, subBlock = repcap.SubBlock.Default, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param filter_type

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.4 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.frequency.clone()
```

Subgroups

6.17.7.6.4.1 Start

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:FREQuency:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>[:FREQuency]:STARt
value: float = driver.sense.spectrum.range.frequency.start.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

frequency: No help available

set(frequency: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>[:FREQuency]:STARt
driver.sense.spectrum.range.frequency.start.set(frequency = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param frequency

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.4.2 Stop**SCPI Commands**

SENSe:ESpectrum<SubBlock>:RANge<RangePy>:FREQuency:STOP

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>[:FREQuency]:STOP value: float = driver.sense.espectrum.range.frequency.stop.get(subBlock = ↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)</pre>

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

frequency: No help available

set(frequency: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>[:FREQuency]:STOP driver.sense.espectrum.range.frequency.stop.set(frequency = 1.0, subBlock = ↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)</pre>

No command help available

param frequency

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.5 InputPy

class InputPyCls

InputPy commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.inputPy.clone()
```

Subgroups

6.17.7.6.5.1 Attenuation

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:INPut:ATTenuation
```

class AttenuationCls

Attenuation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:INPut:ATTenuation
value: float = driver.sense.spectrum.range.inputPy.attenuation.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

attenuation: No help available

set(attenuation: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:INPut:ATTenuation
driver.sense.spectrum.range.inputPy.attenuation.set(attenuation = 1.0,
↳subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param attenuation

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.inputPy.attenuation.clone()
```

Subgroups**6.17.7.6.5.2 Auto****SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPut:ATTenuation:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:INPut:ATTenuation:AUTO
driver.sense.spectrum.range.inputPy.attenuation.auto.set(state = False,
↳ subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.5.3 Gain**class GainCls**

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.inputPy.gain.clone()
```

Subgroups

6.17.7.6.5.4 State

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:INPut:GAIN:STATe
value: bool = driver.sense.spectrum.range.inputPy.gain.state.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:INPut:GAIN:STATe
driver.sense.spectrum.range.inputPy.gain.state.set(state = False, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.5.5 Value

SCPI Commands

SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPut:GAIN:VALue

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:INPut:GAIN[:VALue]
value: float = driver.sense.espectrum.range.inputPy.gain.value.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

gain: No help available

set(gain: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:INPut:GAIN[:VALue]
driver.sense.espectrum.range.inputPy.gain.value.set(gain = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param gain

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.6 Insert

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:INSert
```

class InsertCls

Insert commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → TimeOrder

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:INSert
value: enums.TimeOrder = driver.sense.espectrum.range.insert.get(subBlock = ↵
↵repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

mode: No help available

set(mode: TimeOrder, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:INSert
driver.sense.espectrum.range.insert.set(mode = enums.TimeOrder.AFTER, subBlock ↵
↵= repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param mode

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.7 Limit<LimitIx>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.spectrum.range.limit.repcap_limitIx_get()
driver.sense.spectrum.range.limit.repcap_limitIx_set(repcap.LimitIx.Nr1)
```

class LimitCls

Limit commands group definition. 9 total commands, 3 Subgroups, 0 group commands Repeated Capability: LimitIx, default value after init: LimitIx.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.limit.clone()
```

Subgroups

6.17.7.6.7.1 Absolute

class AbsoluteCls

Absolute commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.limit.absolute.clone()
```

Subgroups

6.17.7.6.7.2 Start

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:ABSolute:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:ABSolute:STARt
value: float = driver.sense.spectrum.range.limit.absolute.start.get(subBlock =
↳ repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳ LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:ABSolute:START
driver.sense.espectrum.range.limit.absolute.start.set(level = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.3 Stop

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:ABSolute:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:ABSolute:STOP
value: float = driver.sense.espectrum.range.limit.absolute.stop.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGE<ri>:LIMit<li>:ABSolute:STOP
driver.sense.espectrum.range.limit.absolute.stop.set(level = 1.0, subBlock = ↵
↵repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↵LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.4 Relative

class RelativeCls

Relative commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.range.limit.relative.clone()
```

Subgroups

6.17.7.6.7.5 Start

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:RELative:STARt
```

class StartCls

Start commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STARt
value: float = driver.sense.espectrum.range.limit.relative.start.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STARt
driver.sense.espectrum.range.limit.relative.start.set(level = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.espectrum.range.limit.relative.start.clone()
```

Subgroups

6.17.7.6.7.6 Abs

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:RELative:STARt:ABS
```

class AbsCls

Abs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STARt:ABS
value: float = driver.sense.espectrum.range.limit.relative.start.abs.
↳ get(subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default,
↳ limitIx = repcap.LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STARt:ABS
driver.sense.espectrum.range.limit.relative.start.abs.set(level = 1.0, subBlock_
↳ repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳ LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.7 Function**SCPI Commands**

SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:RELative:STARt:FUNCTion

class FunctionCls

Function commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → FunctionA

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit:RELative:STARt:FUNCTion value: enums.FunctionA = driver.sense.espectrum.range.limit.relative.start. ↪ function.get(subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy. ↪ Default, limitIx = repcap.LimitIx.Default)</pre>
--

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

function: No help available

set(function: FunctionA, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit:RELative:STARt:FUNCTion driver.sense.espectrum.range.limit.relative.start.function.set(function = enums. ↪ FunctionA.MAX, subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy. ↪ Default, limitIx = repcap.LimitIx.Default)</pre>

No command help available

param function

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.8 Stop**SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:RELative:STOP
```

class StopCls

Stop commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STOP
value: float = driver.sense.spectrum.range.limit.relative.stop.get(subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STOP
driver.sense.spectrum.range.limit.relative.stop.set(level = 1.0, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.limit.relative.stop.clone()
```

Subgroups**6.17.7.6.7.9 Abs****SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIx>:RELative:STOP:ABS
```

class AbsCls

Abs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:LIMit<li>:RELative:STOP:ABS
value: float = driver.sense.spectrum.range.limit.relative.stop.abs.
↳ get(subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default,
↳ limitIx = repcap.LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'ESpectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

level: No help available

set(level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:LIMit<li>:RELative:STOP:ABS
driver.sense.spectrum.range.limit.relative.stop.abs.set(level = 1.0, subBlock.
↳ = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳ LimitIx.Default)
```

No command help available

param level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.10 Function

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:RELative:STOP:FUNctIon
```

class FunctionCls

Function commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → FunctionA

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STOP:FUNctIon
value: enums.FunctionA = driver.sense.espectrum.range.limit.relative.stop.
↪ function.get(subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.
↪ Default, limitIx = repcap.LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

function: No help available

set(function: FunctionA, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:RELative:STOP:FUNctIon
driver.sense.espectrum.range.limit.relative.stop.function.set(function = enums.
↪ FunctionA.MAX, subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.
↪ Default, limitIx = repcap.LimitIx.Default)
```


No command help available

param function

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.17.7.6.7.11 State

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:LIMit<LimitIx>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → LimitState

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:STATe
value: enums.LimitState = driver.sense.espectrum.range.limit.state.get(subBlock,
↳= repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx = repcap.
↳LimitIx.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

state: No help available

set(state: LimitState, subBlock=SubBlock.Default, rangePy=RangePy.Default, limitIx=LimitIx.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:LIMit<li>:STATe
driver.sense.espectrum.range.limit.state.set(state = enums.LimitState.ABSolute,
↳subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default, limitIx
↳= repcap.LimitIx.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

param limitIx

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Limit’)

6.17.7.6.8 MlCalc

SCPI Commands

`SENSe:ESpectrum<SubBlock>:RANge<RangePy>:MLCalc`

class MlCalcCls

MlCalc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*subBlock*=*SubBlock.Default*, *rangePy*=*RangePy.Default*) → FunctionB

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:MLCalc
value: enums.FunctionB = driver.sense.espectrum.range.mlCalc.get(subBlock = ↵
↵repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

return

function: No help available

set(*function*: *FunctionB*, *subBlock*=*SubBlock.Default*, *rangePy*=*RangePy.Default*) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:MLCalc
driver.sense.espectrum.range.mlCalc.set(function = enums.FunctionB.MAX, ↵
↵subBlock = repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param function

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.9 RefLevel**SCPI Commands**

SENSe:ESpectrum<SubBlock>:RANge<RangePy>:RLEVel

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:RLEVel value: float = driver.sense.espectrum.range.refLevel.get(subBlock = repcap. ↳ SubBlock.Default, rangePy = repcap.RangePy.Default)</pre>

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

ref_level: No help available

set(ref_level: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

<pre># SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:RLEVel driver.sense.espectrum.range.refLevel.set(ref_level = 1.0, subBlock = repcap. ↳ SubBlock.Default, rangePy = repcap.RangePy.Default)</pre>

No command help available

param ref_level

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.10 Sweep

class SweepCls

Sweep commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.sweep.clone()
```

Subgroups

6.17.7.6.10.1 Time

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RANge<RangePy>:SWEep:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:SWEep:TIME
value: float = driver.sense.spectrum.range.sweep.time.get(subBlock = repcap.
↳ SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface ‘Range’)

return

sweep_time: No help available

set(sweep_time: float, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:SWEep:TIME
driver.sense.spectrum.range.sweep.time.set(sweep_time = 1.0, subBlock = repcap.
↳ SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param sweep_time

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.spectrum.range.sweep.time.clone()
```

Subgroups**6.17.7.6.10.2 Auto****SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:SWEep:TIME:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANGe<ri>:SWEep:TIME:AUTO
driver.sense.spectrum.range.sweep.time.auto.set(state = False, subBlock =
↳repcap.SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.6.11 Transducer**SCPI Commands**

```
SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:TRANsducer
```

class TransducerCls

Transducer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default, rangePy=RangePy.Default) → str

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:TRANsdUCer
value: str = driver.sense.espectrum.range.transducer.get(subBlock = repcap.
↳SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

transducer: No help available

set(transducer: str, subBlock=SubBlock.Default, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RANge<ri>:TRANsdUCer
driver.sense.espectrum.range.transducer.set(transducer = '1', subBlock = repcap.
↳SubBlock.Default, rangePy = repcap.RangePy.Default)
```

No command help available

param transducer

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.7.7 Range

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RRANge
```

class RrangeCls

Range commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:RRANge
value: float = driver.sense.espectrum.rrange.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return
 ref_range: No help available

6.17.7.8 Rtype

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:RTYPE
```

class RtypeCls

Rtype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → EspectrumRtype

```
# SCPI: [SENSe]:ESpectrum<sb>:RTYPE
value: enums.EspectrumRtype = driver.sense.espectrum.rtype.get(subBlock = ↵
↵repcap.SubBlock.Default)
```

No command help available

param subBlock
 optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return
 type_py: No help available

set(type_py: EspectrumRtype, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:RTYPE
driver.sense.espectrum.rtype.set(type_py = enums.EspectrumRtype.CPOWer, ↵
↵subBlock = repcap.SubBlock.Default)
```

No command help available

param type_py
 No help available

param subBlock
 optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

6.17.7.9 Scenter

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:SCENTER
```

class ScenterCls

Scenter commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:SCENter
value: float = driver.sense.espectrum.scenter.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return

frequency: No help available

set(frequency: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:SCENter
driver.sense.espectrum.scenter.set(frequency = 1.0, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param frequency

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

6.17.7.10 Scout

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:SCount
```

class ScoutCls

Scout commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:ESpectrum<sb>:SCount
value: float = driver.sense.espectrum.scount.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Espectrum’)

return

subblocks: No help available

set(subblocks: float, subBlock=SubBlock.Default) → None


```
# SCPI: [SENSe]:ESpectrum<sb>:SCount
driver.sense.espectrum.scount.set(subblocks = 1.0, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subblocks

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.7.11 Ssetup

SCPI Commands

```
SENSe:ESpectrum<SubBlock>:SSETup
```

class SsetupCls

Ssetup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → bool

```
# SCPI: [SENSe]:ESpectrum<sb>:SSETup
value: bool = driver.sense.espectrum.ssetup.get(subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:ESpectrum<sb>:SSETup
driver.sense.espectrum.ssetup.set(state = False, subBlock = repcap.SubBlock.
↳Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Espectrum')

6.17.8 FilterPy<FilterPy>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.filterPy.repcap_filterPy_get()
driver.sense.filterPy.repcap_filterPy_set(repcap.FilterPy.Nr1)
```

class FilterPyCls

FilterPy commands group definition. 13 total commands, 6 Subgroups, 0 group commands Repeated Capability: FilterPy, default value after init: FilterPy.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.clone()
```

Subgroups

6.17.8.1 Aoff

SCPI Commands

```
SENSe:FILTer<FilterPy>:AOFF
```

class AoffCls

Aoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:AOFF
driver.sense.filterPy.aoff.set(filterPy = repcap.FilterPy.Default)
```

No command help available

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

set_with_opc(filterPy=FilterPy.Default, opc_timeout_ms: int = -1) → None

6.17.8.2 Aweighted

class AweightedCls

Aweighted commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.aweighted.clone()
```

Subgroups

6.17.8.2.1 State

SCPI Commands

```
SENSe:FILTer<FilterPy>:AWEighted:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:AWEighted[:STATe]
value: bool = driver.sense.filterPy.aweighted.state.get(filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the ‘A’ weighting filter for the specified evaluation. For details on weighting filters, see ‘Weighting’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:AWEighted[:STATe]
driver.sense.filterPy.aweighted.state.set(state = False, filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the ‘A’ weighting filter for the specified evaluation. For details on weighting filters, see ‘Weighting’.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

6.17.8.3 Ccir

class CcirCls

Ccir commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.ccir.clone()
```

Subgroups

6.17.8.3.1 Unweighted

class UnweightedCls

Unweighted commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.ccir.unweighted.clone()
```

Subgroups

6.17.8.3.1.1 State

SCPI Commands

```
SENSe:FILTer<FilterPy>:CCIR:UNWeighted:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:CCIR[:UNWeighted][:STATe]
value: bool = driver.sense.filterPy.ccir.unweighted.state.get(filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the unweighted CCIR filter in the specified window. For details on weighting filters, see ‘Weighting’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTER<n>:CCIR[:UNWeighted][:STATe]
driver.sense.filterPy.ccir.unweighted.state.set(state = False, filterPy =
↳repcap.FilterPy.Default)
```

This command activates/deactivates the unweighted CCIR filter in the specified window. For details on weighting filters, see ‘Weighting’.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

6.17.8.3.2 Weighted

class WeightedCls

Weighted commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.ccir.weighted.clone()
```

Subgroups

6.17.8.3.2.1 State

SCPI Commands

```
SENSe:FILTer<FilterPy>:CCIR:WEIGHted:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:CCIR:WEIGHted[:STATe]
value: bool = driver.sense.filterPy.ccir.weighted.state.get(filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the weighted CCIR filter for the specified evaluation. For details on weighting filters, see ‘Weighting’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:CCIR:WEIGHted[:STATe]
driver.sense.filterPy.ccir.weighted.state.set(state = False, filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the weighted CCIR filter for the specified evaluation. For details on weighting filters, see ‘Weighting’.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

6.17.8.4 Demphasis

class DemphasisCls

Demphasis commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.demphasis.clone()
```

Subgroups

6.17.8.4.1 State

SCPI Commands

```
SENSe:FILTer<FilterPy>:DEMPhasis:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:DEMPhasis[:STATe]
value: bool = driver.sense.filterPy.demphasis.state.get(filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the selected deemphasis for the specified evaluation. For details about deemphasis refer to ‘Deemphasis’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTER<n>:DEMPHasis[:STATE]
driver.sense.filterPy.demphasis.state.set(state = False, filterPy = repcap.
↳FilterPy.Default)
```

This command activates/deactivates the selected deemphasis for the specified evaluation. For details about deemphasis refer to ‘Deemphasis’.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

6.17.8.4.2 Tconstant

SCPI Commands

```
SENSe:FILTer<FilterPy>:DEMPHasis:TCONstant
```

class TconstantCls

Tconstant commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:DEMPHasis:TCONstant
value: float = driver.sense.filterPy.demphasis.tconstant.get(filterPy = repcap.
↳FilterPy.Default)
```

This command selects the deemphasis for the specified evaluation. For details on deemphasis refer to ‘Deemphasis’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

value: 25 us | 50 us | 75 us | 750 us Unit: S

set(value: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:DEMPHasis:TCONstant
driver.sense.filterPy.demphasis.tconstant.set(value = 1.0, filterPy = repcap.
↳FilterPy.Default)
```

This command selects the deemphasis for the specified evaluation. For details on deemphasis refer to ‘Deemphasis’.

param value

25 us | 50 us | 75 us | 750 us Unit: S

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.5 Hpass**class HpassCls**

Hpass commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.hpass.clone()
```

Subgroups**6.17.8.5.1 Frequency****class FrequencyCls**

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.hpass.frequency.clone()
```

Subgroups**6.17.8.5.1.1 Absolute****SCPI Commands**

```
SENSe:FILTer<FilterPy>:HPASs:FREQuency:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:HPASs:FREQuency[:ABSolute]
value: float = driver.sense.filterPy.hpass.frequency.absolute.get(filterPy =
↳repcap.FilterPy.Default)
```

This command selects the high pass filter type for the specified evaluation. For details on the high pass filters, refer to 'High Pass'.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

return

frequency: 20 Hz | 50 Hz | 300 Hz Unit: Hz

set(frequency: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTER<n>:HPASs:FREQuency[:ABSolute]
driver.sense.filterPy.hpass.frequency.absolute.set(frequency = 1.0, filterPy =
↳repcap.FilterPy.Default)
```

This command selects the high pass filter type for the specified evaluation. For details on the high pass filters, refer to 'High Pass'.

param frequency

20 Hz | 50 Hz | 300 Hz Unit: Hz

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.5.1.2 Manual**SCPI Commands**

```
SENSe:FILTer<FilterPy>:HPASs:FREQuency:MANual
```

class ManualCls

Manual commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:HPASs:FREQuency:MANual
value: float = driver.sense.filterPy.hpass.frequency.manual.get(filterPy =
↳repcap.FilterPy.Default)
```

This command selects the cutoff frequency of the high pass filter for the specified evaluation. For details on the high pass filters, refer to 'High Pass'.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

return

frequency: numeric value Range: 0 to 3 MHz, Unit: HZ

set(frequency: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:HPASs:FREQuency:MANual
driver.sense.filterPy.hpass.frequency.manual.set(frequency = 1.0, filterPy =
↳repcap.FilterPy.Default)
```

This command selects the cutoff frequency of the high pass filter for the specified evaluation. For details on the high pass filters, refer to 'High Pass'.

param frequency

numeric value Range: 0 to 3 MHz, Unit: HZ

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.5.2 State**SCPI Commands**

SENSe:FILTer<FilterPy>:HPASs:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:HPASs[:STATe]
value: bool = driver.sense.filterPy.hpass.state.get(filterPy = repcap.FilterPy.
↳Default)
```

This command activates/deactivates the selected high pass filter for the specified evaluation. For details on the high pass filter, refer to 'High Pass'.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

return

state: ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:HPASs[:STATe]
driver.sense.filterPy.hpass.state.set(state = False, filterPy = repcap.FilterPy.
↳Default)
```

This command activates/deactivates the selected high pass filter for the specified evaluation. For details on the high pass filter, refer to 'High Pass'.

param state

ON | OFF | 0 | 1 OFF | 0 Switches the function off ON | 1 Switches the function on

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.6 Lpass

class LpassCls

Lpass commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.lpass.clone()
```

Subgroups

6.17.8.6.1 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.filterPy.lpass.frequency.clone()
```

Subgroups

6.17.8.6.1.1 Absolute

SCPI Commands

```
SENSe:FILTer<FilterPy>:LPASs:FREQuency:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency[:ABSolute]
value: float = driver.sense.filterPy.lpass.frequency.absolute.get(filterPy =
↳repcap.FilterPy.Default)
```

This command selects the absolute low pass filter type for the specified evaluation For details on the low pass filter, refer to ‘Low Pass’.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

frequency: 3kHz | 15kHz | 150kHz Unit: HZ

set(frequency: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency[:ABSolute]
driver.sense.filterPy.lpass.frequency.absolute.set(frequency = 1.0, filterPy =
↳repcap.FilterPy.Default)
```

This command selects the absolute low pass filter type for the specified evaluation. For details on the low pass filter, refer to ‘Low Pass’.

param frequency

3kHz | 15kHz | 150kHz Unit: HZ

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

6.17.8.6.1.2 Manual

SCPI Commands

```
SENSe:FILTer<FilterPy>:LPASs:FREQuency:MANual
```

class ManualCls

Manual commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency:MANual
value: float = driver.sense.filterPy.lpass.frequency.manual.get(filterPy =
↳repcap.FilterPy.Default)
```

This command defines the cutoff frequency of the low pass filter you can use to measure small carrier frequencies.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on low pass filter ([SENSe:]FILTer:LPASs[:STATe]) .

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘FilterPy’)

return

frequency: numeric value Unit: Hz

set(frequency: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency:MANual
driver.sense.filterPy.lpass.frequency.manual.set(frequency = 1.0, filterPy =
↳repcap.FilterPy.Default)
```

This command defines the cutoff frequency of the low pass filter you can use to measure small carrier frequencies.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on low pass filter ([SENSe:]FILTer:LPASs[:STATe]) .

param frequency

numeric value Unit: Hz

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.6.1.3 Relative**SCPI Commands**

SENSe:FILTer<FilterPy>:LPASs:FREQuency:RELative

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → float

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency:RELative
value: float = driver.sense.filterPy.lpass.frequency.relative.get(filterPy =
↳repcap.FilterPy.Default)
```

This command selects the relative low pass filter type for the specified evaluation For details on the low pass filter, refer to 'Low Pass'.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

return

frequency: 5PCT | 10PCT | 25PCT Unit: PCT

set(frequency: float, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:LPASs:FREQuency:RELative
driver.sense.filterPy.lpass.frequency.relative.set(frequency = 1.0, filterPy =
↳repcap.FilterPy.Default)
```

This command selects the relative low pass filter type for the specified evaluation For details on the low pass filter, refer to 'Low Pass'.

param frequency

5PCT | 10PCT | 25PCT Unit: PCT

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.8.6.2 State

SCPI Commands

```
SENSe:FILTer<FilterPy>:LPASs:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(filterPy=FilterPy.Default) → bool

```
# SCPI: [SENSe]:FILTer<n>:LPASs[:STATe]
value: bool = driver.sense.filterPy.lpass.state.get(filterPy = repcap.FilterPy.
↳Default)
```

This command turns a low pass filter for measurements on small carrier frequencies on and off.

INTRO_CMD_HELP: Effects of using the low pass filter:

- Auto search feature is turned off ([SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]) .
- Signal count is turned off ([SENSe:]ADJust:CONFigure:FREQuency:COUNt) .
- The stop offset is limited to 20 % of the filter cut-off frequency ([SENSe:]FREQuency:STOP) .
- DC coupling should be used for measurements on carrier frequencies below 1 MHz (method RsFswp.Applications.K30_NoiseFigure.InputPy.Coupling.set) .

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

return

state: ON | OFF | 1 | 0 When you turn on the filter, you can define a cutoff frequency with [SENSe:]FILTer:LPASs:FREQuency:MANual.

set(state: bool, filterPy=FilterPy.Default) → None

```
# SCPI: [SENSe]:FILTer<n>:LPASs[:STATe]
driver.sense.filterPy.lpass.state.set(state = False, filterPy = repcap.FilterPy.
↳Default)
```

This command turns a low pass filter for measurements on small carrier frequencies on and off.

INTRO_CMD_HELP: Effects of using the low pass filter:

- Auto search feature is turned off ([SENSe:]ADJust:CONFigure:FREQuency:AUTosearch[:STATe]) .
- Signal count is turned off ([SENSe:]ADJust:CONFigure:FREQuency:COUNt) .
- The stop offset is limited to 20 % of the filter cut-off frequency ([SENSe:]FREQuency:STOP) .
- DC coupling should be used for measurements on carrier frequencies below 1 MHz (method RsFswp.Applications.K30_NoiseFigure.InputPy.Coupling.set) .

param state

ON | OFF | 1 | 0 When you turn on the filter, you can define a cutoff frequency with [SENSe:]FILTer:LPASs:FREQuency:MANual.

param filterPy

optional repeated capability selector. Default value: Nr1 (settable in the interface 'FilterPy')

6.17.9 Frequency

class FrequencyCls

Frequency commands group definition. 11 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.frequency.clone()
```

Subgroups

6.17.9.1 Annotation

SCPI Commands

```
SENSe:FREQuency:ANNotation
```

class AnnotationCls

Annotation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AnnotationMode

```
# SCPI: [SENSe]:FREQuency:ANNotation
value: enums.AnnotationMode = driver.sense.frequency.annotation.get()
```

No command help available

return

mode: No help available

set(mode: AnnotationMode) → None

```
# SCPI: [SENSe]:FREQuency:ANNotation
driver.sense.frequency.annotation.set(mode = enums.AnnotationMode.CSPan)
```

No command help available

param mode

No help available

6.17.9.2 Center

SCPI Commands

`SENSe:FREQuency:CENTer`

class CenterCls

Center commands group definition. 5 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:CENTer
value: float = driver.sense.frequency.center.get()
```

This command defines the center frequency.

return

frequency: The allowed range and fmax is specified in the data sheet. Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:CENTer
driver.sense.frequency.center.set(frequency = 1.0)
```

This command defines the center frequency.

param frequency

The allowed range and fmax is specified in the data sheet. Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.frequency.center.clone()
```

Subgroups

6.17.9.2.1 Step

SCPI Commands

`SENSe:FREQuency:CENTer:STEP`

class StepCls

Step commands group definition. 4 total commands, 2 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP
value: float = driver.sense.frequency.center.step.get()
```

This command defines the center frequency step size. You can increase or decrease the center frequency quickly in fixed steps using the SENS:FREQ UP AND SENS:FREQ DOWN commands, see [SENSe:]FREQuency:CENTer.

return

frequency_step: No help available

set(frequency_step: float) → None

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP
driver.sense.frequency.center.step.set(frequency_step = 1.0)
```

This command defines the center frequency step size. You can increase or decrease the center frequency quickly in fixed steps using the SENS:FREQ UP AND SENS:FREQ DOWN commands, see [SENSe:]FREQuency:CENTer.

param frequency_step

fmax is specified in the data sheet. Range: 1 to fMAX, Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.frequency.center.step.clone()
```

Subgroups

6.17.9.2.1.1 Auto

SCPI Commands

```
SENSe:FREQuency:CENTer:STEP:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:AUTO
driver.sense.frequency.center.step.auto.set(state = False)
```

This command couples or decouples the center frequency step size to the span. In time domain (zero span) measurements, the center frequency is coupled to the RBW.

param state

ON | OFF | 0 | 1

6.17.9.2.1.2 Link

SCPI Commands

```
SENSe:FREQuency:CENTer:STEP:LINK
```

class LinkCls

Link commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → FrequencyCouplingLinkA

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:LINK
value: enums.FrequencyCouplingLinkA = driver.sense.frequency.center.step.link.
↪get()
```

This command selects the frequency step size mode.

return
coupling_type: No help available

set(coupling_type: FrequencyCouplingLinkA) → None

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:LINK
driver.sense.frequency.center.step.link.set(coupling_type = enums.
↪FrequencyCouplingLinkA.OFF)
```

This command selects the frequency step size mode.

param coupling_type
The step size is either a function of the span (x % of the span) or an absolute value. OFF Step size is a custom percentage of the span or an absolute custom value in Hz. You can define a percentage with [SENSe:]FREQuency:CENTer:STEP:LINK:FACTOR. You can define a custom value with [SENSe:]FREQuency:CENTer:STEP:SPAN Step size is 10 % of the span.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.frequency.center.step.link.clone()
```

Subgroups

6.17.9.2.1.3 Factor

SCPI Commands

```
SENSe:FREQuency:CENTer:STEP:LINK:FACTOR
```

class FactorCls

Factor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:CENTer:STEP:LINK:FACTOR
value: float = driver.sense.frequency.center.step.link.factor.get()
```

This command defines a step size factor if the center frequency step size is coupled to the span or the resolution bandwidth.

return
link_factor: No help available

set(*link_factor: float*) → None

```
# SCPI: [SENSe]:FREQUENCY:CENTer:STEP:LINK:FACTOR
driver.sense.frequency.center.step.link.factor.set(link_factor = 1.0)
```

This command defines a step size factor if the center frequency step size is coupled to the span or the resolution bandwidth.

param link_factor
1 to 100 PCT Unit: PCT

6.17.9.3 Offset

SCPI Commands

SENSe:FREQUENCY:OFFSet

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQUENCY:OFFSet
value: float = driver.sense.frequency.offset.get()
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type 'frequency' will be corrected for this shift numerically by the application. See also 'Frequency Offset'. Note: In MSRA mode, the setting command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

return
frequency: No help available

set(*frequency: float*) → None

```
# SCPI: [SENSe]:FREQUENCY:OFFSet
driver.sense.frequency.offset.set(frequency = 1.0)
```

This command defines a frequency offset. If this value is not 0 Hz, the application assumes that the input signal was frequency shifted outside the application. All results of type 'frequency' will be corrected for this shift numerically by the application. See also 'Frequency Offset'. Note: In MSRA mode, the setting command is only available for the MSRA primary application. For MSRA secondary applications, only the query command is available.

param frequency
Range: -1 THz to 1 THz, Unit: HZ

6.17.9.4 Span

SCPI Commands

`SENSe:FREQuency:SPAN`

class SpanCls

Span commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:SPAN
value: float = driver.sense.frequency.span.get()
```

This command defines the frequency span.

return

span: No help available

set(span: float) → None

```
# SCPI: [SENSe]:FREQuency:SPAN
driver.sense.frequency.span.set(span = 1.0)
```

This command defines the frequency span.

param span

numeric value Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.frequency.span.clone()
```

Subgroups

6.17.9.4.1 Full

SCPI Commands

`SENSe:FREQuency:SPAN:FULL`

class FullCls

Full commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:FREQuency:SPAN:FULL
driver.sense.frequency.span.full.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:FREQuency:SPAN:FULL
driver.sense.frequency.span.full.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.9.5 Start

SCPI Commands

```
SENSe:FREQuency:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:START
value: float = driver.sense.frequency.start.get()
```

CW, pulsed and VCO measurements: This command defines the start frequency offset of the measurement range. Transient measurement: This command defines the start frequency of the transient measurement. Frequency stability measurement: The start frequency offset is coupled to the stop time of the frequency stability measurement ([SENSe:]TIME:STOP) . If you change the start frequency offset, the stop time is adjusted accordingly. For example, 1 mHz corresponds to 1000 s.

return

frequency: Offset frequencies in half decade steps. Range: See data sheet , Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:START
driver.sense.frequency.start.set(frequency = 1.0)
```

CW, pulsed and VCO measurements: This command defines the start frequency offset of the measurement range. Transient measurement: This command defines the start frequency of the transient measurement. Frequency stability measurement: The start frequency offset is coupled to the stop time of the frequency stability measurement ([SENSe:]TIME:STOP) . If you change the start frequency offset, the stop time is adjusted accordingly. For example, 1 mHz corresponds to 1000 s.

param frequency

Offset frequencies in half decade steps. Range: See data sheet , Unit: Hz

6.17.9.6 Stop

SCPI Commands

`SENSe:FREQuency:STOP`

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:FREQuency:STOP
value: float = driver.sense.frequency.stop.get()
```

CW, pulsed and VCO measurements: This command defines the stop frequency offset of the measurement range. Transient measurement: This command defines the stop frequency of the transient measurement. Frequency stability measurement: The stop frequency offset is coupled to the start time of the frequency stability measurement ([SENSe:]TIME:START) . If you change the stop frequency offset, the start time is adjusted accordingly. For example, 1 MHz corresponds to 1 ms.

return

frequency: Offset frequencies in half decade steps. Range: See data sheet , Unit: Hz

set(frequency: float) → None

```
# SCPI: [SENSe]:FREQuency:STOP
driver.sense.frequency.stop.set(frequency = 1.0)
```

CW, pulsed and VCO measurements: This command defines the stop frequency offset of the measurement range. Transient measurement: This command defines the stop frequency of the transient measurement. Frequency stability measurement: The stop frequency offset is coupled to the start time of the frequency stability measurement ([SENSe:]TIME:START) . If you change the stop frequency offset, the start time is adjusted accordingly. For example, 1 MHz corresponds to 1 ms.

param frequency

Offset frequencies in half decade steps. Range: See data sheet , Unit: Hz

6.17.10 Iq

class IqCls

Iq commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.iq.clone()
```

Subgroups

6.17.10.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.iq.bandwidth.clone()
```

Subgroups

6.17.10.1.1 Mode

SCPI Commands

```
SENSe:IQ:BWIDth:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → IqBandwidthMode

```
# SCPI: [SENSe]:IQ:BWIDth:MODE
value: enums.IqBandwidthMode = driver.sense.iq.bandwidth.mode.get()
```

This command defines how the resolution bandwidth is determined.

return

mode: AUTO | MANual | FFT AUTO (Default) The RBW is determined automatically depending on the sample rate and record length. MANual The user-defined RBW is used and the (FFT) window length (and possibly the sample rate) are adapted accordingly. The RBW is defined using the [SENSe:]IQ:BWIDth:RESolution command. FFT The RBW is determined by the FFT parameters.

set(mode: IqBandwidthMode) → None

```
# SCPI: [SENSe]:IQ:BWIDth:MODE
driver.sense.iq.bandwidth.mode.set(mode = enums.IqBandwidthMode.AUTO)
```

This command defines how the resolution bandwidth is determined.

param mode

AUTO | MANual | FFT AUTO (Default) The RBW is determined automatically depending on the sample rate and record length. MANual The user-defined RBW is used and the (FFT) window length (and possibly the sample rate) are adapted accordingly. The RBW is defined using the [SENSe:]IQ:BWIDth:RESolution command. FFT The RBW is determined by the FFT parameters.

6.17.10.1.2 Resolution

SCPI Commands

`SENSe:IQ:BWIDth:RESolution`

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:IQ:BWIDth:RESolution
value: float = driver.sense.iq.bandwidth.resolution.get()
```

This command defines the resolution bandwidth manually if [SENSe:]IQ:BWIDth:MODE is set to MAN. Defines the resolution bandwidth. The available RBW values depend on the sample rate and record length. For details see ‘Frequency resolution of FFT results - RBW’.

return

bandwidth: refer to data sheet Unit: HZ

set(bandwidth: float) → None

```
# SCPI: [SENSe]:IQ:BWIDth:RESolution
driver.sense.iq.bandwidth.resolution.set(bandwidth = 1.0)
```

This command defines the resolution bandwidth manually if [SENSe:]IQ:BWIDth:MODE is set to MAN. Defines the resolution bandwidth. The available RBW values depend on the sample rate and record length. For details see ‘Frequency resolution of FFT results - RBW’.

param bandwidth

refer to data sheet Unit: HZ

6.17.11 ListPy

class ListPyCls

ListPy commands group definition. 26 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.clone()
```


Subgroups

6.17.11.1 Power

class PowerCls

Power commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.power.clone()
```

Subgroups

6.17.11.1.1 Result

SCPI Commands

```
SENSe:LIST:POWer:RESt
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:LIST:POWer:RESt
value: float = driver.sense.listPy.power.result.get()
```

No command help available

```
return
    power_level: No help available
```

6.17.11.1.2 Sequence

SCPI Commands

```
SENSe:LIST:POWer:SEquence
```

class SequenceCls

Sequence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SequenceStruct

Structure for setting input parameters. Fields:

- Frequency: List[float]: No parameter help available
- Ref_Level: List[float]: No parameter help available
- Rfattenuation: List[float]: No parameter help available
- Filter_Type: List[float or bool]: No parameter help available

- Rbw: List[enums.FilterTypeK91]: No parameter help available
- Vbw: List[float]: No parameter help available
- Meas_Time: List[float]: No parameter help available
- Trigger_Level: List[float]: No parameter help available
- Power_Level: List[float]: No parameter help available

get() → SequenceStruct

```
# SCPI: [SENSe]:LIST:POWer[:SEquence]
value: SequenceStruct = driver.sense.listPy.power.sequence.get()
```

No command help available

return

structure: for return value, see the help for SequenceStruct structure arguments.

set(structure: SequenceStruct) → None

```
# SCPI: [SENSe]:LIST:POWer[:SEquence]
structure = driver.sense.listPy.power.sequence.SequenceStruct()
structure.Frequency: List[float] = [1.1, 2.2, 3.3]
structure.Ref_Level: List[float] = [1.1, 2.2, 3.3]
structure.Rfattenuation: List[float] = [1.1, 2.2, 3.3]
structure.Filter_Type: List[float or bool] = [1.1, True, 2.2, False, 3.3]
structure.Rbw: List[enums.FilterTypeK91] = [FilterTypeK91.CFILTter, ↵
↵FilterTypeK91.RRC]
structure.Vbw: List[float] = [1.1, 2.2, 3.3]
structure.Meas_Time: List[float] = [1.1, 2.2, 3.3]
structure.Trigger_Level: List[float] = [1.1, 2.2, 3.3]
structure.Power_Level: List[float] = [1.1, 2.2, 3.3]
driver.sense.listPy.power.sequence.set(structure)
```

No command help available

param structure

for set value, see the help for SequenceStruct structure arguments.

6.17.11.1.3 Set

SCPI Commands

```
SENSe:LIST:POWer:SET
```

class SetCls

Set commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetStruct

Structure for setting input parameters. Fields:

- State_Peak: bool: No parameter help available
- State_Rms: bool: No parameter help available
- State_Avg: bool: No parameter help available

- Trigger_Source: enums.TriggerSourceListPower: No parameter help available
- Trigger_Slope: enums.SlopeType: No parameter help available
- Trigger_Offset: float: No parameter help available
- Gate_Length: float: No parameter help available

get() → SetStruct

```
# SCPI: [SENSe]:LIST:POWer:SET
value: SetStruct = driver.sense.listPy.power.set.get()
```

No command help available

return

structure: for return value, see the help for SetStruct structure arguments.

set(structure: SetStruct) → None

```
# SCPI: [SENSe]:LIST:POWer:SET
structure = driver.sense.listPy.power.set.SetStruct()
structure.State_Peak: bool = False
structure.State_Rms: bool = False
structure.State_Avg: bool = False
structure.Trigger_Source: enums.TriggerSourceListPower = enums.
↳ TriggerSourceListPower.EXT2
structure.Trigger_Slope: enums.SlopeType = enums.SlopeType.NEGative
structure.Trigger_Offset: float = 1.0
structure.Gate_Length: float = 1.0
driver.sense.listPy.power.set.set(structure)
```

No command help available

param structure

for set value, see the help for SetStruct structure arguments.

6.17.11.1.4 State

SCPI Commands

```
SENSe:LIST:POWer:STaTe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → OffState

```
# SCPI: [SENSe]:LIST:POWer:STaTe
value: enums.OffState = driver.sense.listPy.power.state.get()
```

No command help available

return

state: No help available

set(state: OffState) → None

```
# SCPI: [SENSe]:LIST:POWer:STATe
driver.sense.listPy.power.state.set(state = enums.OffState.OFF)
```

No command help available

param state

No help available

6.17.11.2 Range<RangePy>

RepCap Settings

```
# Range: Ix1 .. Ix64
rc = driver.sense.listPy.range.repcap_rangePy_get()
driver.sense.listPy.range.repcap_rangePy_set(repcap.RangePy.Ix1)
```

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:DELeTe
```

class RangeCls

Range commands group definition. 21 total commands, 12 Subgroups, 1 group commands Repeated Capability: RangePy, default value after init: RangePy.Ix1

delete(rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:DELeTe
driver.sense.listPy.range.delete(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

delete_with_opc(rangePy=RangePy.Default, opc_timeout_ms: int = -1) → None

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.clone()
```

Subgroups

6.17.11.2.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.bandwidth.clone()
```

Subgroups

6.17.11.2.1.1 Resolution

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:BANDwidth:RESolution
```

class ResolutionCls

Resolution commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANDwidth:RESolution
value: float = driver.sense.listPy.range.bandwidth.resolution.get(rangePy =
↳repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

rbw: No help available

set(rbw: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANDwidth:RESolution
driver.sense.listPy.range.bandwidth.resolution.set(rbw = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rbw

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.1.2 Video

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:BANdwidth:VIDeo`

class VideoCls

Video commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy*=*RangePy.Default*) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANdwidth:VIDeo
value: float = driver.sense.listPy.range.bandwidth.video.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

vbw: No help available

set(*vbw*: float, *rangePy*=*RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BANdwidth:VIDeo
driver.sense.listPy.range.bandwidth.video.set(vbw = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param vbw

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.2 BreakPy

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:BREak`

class BreakPyCls

BreakPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy*=*RangePy.Default*) → bool

```
# SCPI: [SENSe]:LIST:RANGe<ri>:BREak
value: bool = driver.sense.listPy.range.breakPy.get(rangePy = repcap.RangePy.
↳Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: No help available

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANge<ri>:BREak
driver.sense.listPy.range.breakPy.set(state = False, rangePy = repcap.RangePy.
↳Default)
```

No command help available

param state

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.3 Count**SCPI Commands**

```
SENSe:LIST:RANge<RangePy>:COUNt
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANge<ri>:COUNt
value: float = driver.sense.listPy.range.count.get(rangePy = repcap.RangePy.
↳Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

ranges: No help available

6.17.11.2.4 Detector

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:DETECTOR`

class DetectorCls

Detector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → DetectorC

```
# SCPI: [SENSe]:LIST:RANGe<ri>:DETECTOR
value: enums.DetectorC = driver.sense.listPy.range.detector.get(rangePy = ↵
↵repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

detector: No help available

set(detector: DetectorC, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:DETECTOR
driver.sense.listPy.range.detector.set(detector = enums.DetectorC.ACSine, ↵
↵rangePy = repcap.RangePy.Default)
```

No command help available

param detector

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.5 FilterPy

class FilterPyCls

FilterPy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.filterPy.clone()
```


Subgroups

6.17.11.2.5.1 TypePy

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:FILTer:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → FilterTypeC

```
# SCPI: [SENSe]:LIST:RANGe<ri>:FILTer:TYPE
value: enums.FilterTypeC = driver.sense.listPy.range.filterPy.typePy.
↳get(rangePy = repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

filter_type: No help available

set(filter_type: FilterTypeC, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:FILTer:TYPE
driver.sense.listPy.range.filterPy.typePy.set(filter_type = enums.FilterTypeC.
↳CFILter, rangePy = repcap.RangePy.Default)
```

No command help available

param filter_type

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.6 Frequency

class FrequencyCls

Frequency commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.frequency.clone()
```

Subgroups

6.17.11.2.6.1 Start

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:FREQuency:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STARt
value: float = driver.sense.listPy.range.frequency.start.get(rangePy = repcap.
↳RangePy.Default)
```

This command queries the start frequency offset of a half decade.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

frequency: numeric value Unit: Hz

set(frequency: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STARt
driver.sense.listPy.range.frequency.start.set(frequency = 1.0, rangePy = repcap.
↳RangePy.Default)
```

This command queries the start frequency offset of a half decade.

param frequency

numeric value Unit: Hz

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.6.2 Stop

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:FREQuency:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STOP
value: float = driver.sense.listPy.range.frequency.stop.get(rangePy = repcap.
↳RangePy.Default)
```

This command queries the stop frequency offset of a half decade.

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

frequency: numeric value Unit: Hz

set(frequency: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>[:FREQuency]:STOP
driver.sense.listPy.range.frequency.stop.set(frequency = 1.0, rangePy = repcap.
↳RangePy.Default)
```

This command queries the stop frequency offset of a half decade.

param frequency

numeric value Unit: Hz

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.7 InputPy

class InputPyCls

InputPy commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.inputPy.clone()
```

Subgroups

6.17.11.2.7.1 Attenuation

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:INPut:ATTenuation`

class AttenuationCls

Attenuation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:ATTenuation
value: float = driver.sense.listPy.range.inputPy.attenuation.get(rangePy =
↳repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

attenuation: No help available

set(attenuation: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:ATTenuation
driver.sense.listPy.range.inputPy.attenuation.set(attenuation = 1.0, rangePy =
↳repcap.RangePy.Default)
```

No command help available

param attenuation

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.inputPy.attenuation.clone()
```

Subgroups

6.17.11.2.7.2 Auto

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:ATTenuation:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:ATTenuation:AUTO
driver.sense.listPy.range.inputPy.attenuation.auto.set(state = False, rangePy =
↳repcap.RangePy.Default)
```

No command help available

param state

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.7.3 Gain

class GainCls

Gain commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.inputPy.gain.clone()
```

Subgroups

6.17.11.2.7.4 State

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:GAIN:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → bool

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN:STATe
value: bool = driver.sense.listPy.range.inputPy.gain.state.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: No help available

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN:STATe
driver.sense.listPy.range.inputPy.gain.state.set(state = False, rangePy =
↳repcap.RangePy.Default)
```

No command help available

param state

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.7.5 Value

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:INPut:GAIN:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN[:VALue]
value: float = driver.sense.listPy.range.inputPy.gain.value.get(rangePy =
↳repcap.RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

gain: No help available

set(gain: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:INPut:GAIN[:VALue]
driver.sense.listPy.range.inputPy.gain.value.set(gain = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param gain

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.8 Limit

class LimitCls

Limit commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.limit.clone()
```

Subgroups

6.17.11.2.8.1 Start

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:LIMit:START
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:START
value: float = driver.sense.listPy.range.limit.start.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

level: No help available

set(*level: float, rangePy=RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:STARt
driver.sense.listPy.range.limit.start.set(level = 1.0, rangePy = repcap.RangePy.
↳Default)
```

No command help available

param level

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.8.2 State

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:LIMit:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy=RangePy.Default*) → bool

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:STATe
value: bool = driver.sense.listPy.range.limit.state.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

state: No help available

set(*state: bool, rangePy=RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:STATe
driver.sense.listPy.range.limit.state.set(state = False, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param state

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.8.3 Stop

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:LIMit:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:STOP
value: float = driver.sense.listPy.range.limit.stop.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

level: No help available

set(level: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:LIMit:STOP
driver.sense.listPy.range.limit.stop.set(level = 1.0, rangePy = repcap.RangePy.
↳Default)
```

No command help available

param level

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.9 Points

class PointsCls

Points commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.points.clone()
```

Subgroups

6.17.11.2.9.1 Value

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:POINts:VALue`

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy*=*RangePy.Default*) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:POINts[:VALue]
value: float = driver.sense.listPy.range.points.value.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

points: No help available

set(*points*: float, *rangePy*=*RangePy.Default*) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:POINts[:VALue]
driver.sense.listPy.range.points.value.set(points = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param points

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.10 RefLevel

SCPI Commands

`SENSe:LIST:RANGe<RangePy>:RLEVel`

class RefLevelCls

RefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*rangePy*=*RangePy.Default*) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:RLEVel
value: float = driver.sense.listPy.range.refLevel.get(rangePy = repcap.RangePy.
↳Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

ref_level: No help available

set(ref_level: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:RLEVel
driver.sense.listPy.range.refLevel.set(ref_level = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param ref_level

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.11 Sweep

class SweepCls

Sweep commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.sweep.clone()
```

Subgroups

6.17.11.2.11.1 Time

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:SWEEp:TIME
```

class TimeCls

Time commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → float

```
# SCPI: [SENSe]:LIST:RANGe<ri>:SWEEp:TIME
value: float = driver.sense.listPy.range.sweep.time.get(rangePy = repcap.
↳RangePy.Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

return

sweep_time: No help available

set(sweep_time: float, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:SWEep:TIME
driver.sense.listPy.range.sweep.time.set(sweep_time = 1.0, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param sweep_time

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.listPy.range.sweep.time.clone()
```

Subgroups

6.17.11.2.11.2 Auto

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:SWEep:TIME:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:SWEep:TIME:AUTO
driver.sense.listPy.range.sweep.time.auto.set(state = False, rangePy = repcap.
↳RangePy.Default)
```

No command help available

param state

No help available

param rangePy

optional repeated capability selector. Default value: Ix1 (settable in the interface 'Range')

6.17.11.2.12 Transducer

SCPI Commands

```
SENSe:LIST:RANGe<RangePy>:TRANsducer
```

class TransducerCls

Transducer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(rangePy=RangePy.Default) → str

```
# SCPI: [SENSe]:LIST:RANGe<ri>:TRANsducer
value: str = driver.sense.listPy.range.transducer.get(rangePy = repcap.RangePy.
↳Default)
```

No command help available

param rangePy

optional repeated capability selector. Default value: 1x1 (settable in the interface 'Range')

return

transducer: No help available

set(transducer: str, rangePy=RangePy.Default) → None

```
# SCPI: [SENSe]:LIST:RANGe<ri>:TRANsducer
driver.sense.listPy.range.transducer.set(transducer = '1', rangePy = repcap.
↳RangePy.Default)
```

No command help available

param transducer

No help available

param rangePy

optional repeated capability selector. Default value: 1x1 (settable in the interface 'Range')

6.17.11.3 Xadjust

SCPI Commands

```
SENSe:LIST:XADJust
```

class XadjustCls

Xadjust commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: [SENSe]:LIST:XADJust
driver.sense.listPy.xadjust.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:LIST:XADJust
driver.sense.listPy.xadjust.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.12 Mixer

class MixerCls

Mixer commands group definition. 22 total commands, 11 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.clone()
```

Subgroups

6.17.12.1 Bias

class BiasCls

Bias commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.bias.clone()
```

Subgroups

6.17.12.1.1 High

SCPI Commands

```
SENSe:MIXer:BIAS:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:BIAS:HIGH
value: float = driver.sense.mixer.bias.high.get()
```

This command defines the bias current for the high (last) range. (See also ‘Bias current’) . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

return
bias_setting: Unit: A

set(bias_setting: float) → None

```
# SCPI: [SENSe]:MIXer:BIAS:HIGH
driver.sense.mixer.bias.high.set(bias_setting = 1.0)
```

This command defines the bias current for the high (last) range. (See also ‘Bias current’) . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param bias_setting
Unit: A

6.17.12.1.2 Low

SCPI Commands

```
SENSe:MIXer:BIAS:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
value: float = driver.sense.mixer.bias.low.get()
```

This command defines the bias current for the low (first) range. (See also ‘Bias current’) . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

return
bias_setting: Unit: A

set(bias_setting: float) → None

```
# SCPI: [SENSe]:MIXer:BIAS[:LOW]
driver.sense.mixer.bias.low.set(bias_setting = 1.0)
```

This command defines the bias current for the low (first) range. (See also ‘Bias current’) . This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param bias_setting
Unit: A

6.17.12.2 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.frequency.clone()
```

Subgroups

6.17.12.2.1 Handover

SCPI Commands

```
SENSe:MIXer:FREquency:HANdOver
```

class HandoverCls

Handover commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREquency:HANdOver
value: float = driver.sense.mixer.frequency.handover.get()
```

This command defines the frequency at which the mixer switches from one range to the next (if two different ranges are selected) . The handover frequency for each band can be selected freely within the overlapping frequency range. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

return
frequency: Unit: HZ

set(frequency: float) → None

```
# SCPI: [SENSe]:MIXer:FREquency:HANdOver
driver.sense.mixer.frequency.handover.set(frequency = 1.0)
```

This command defines the frequency at which the mixer switches from one range to the next (if two different ranges are selected) . The handover frequency for each band can be selected freely within the overlapping frequency range. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]) .

param frequency
Unit: HZ

6.17.12.2.2 Start

SCPI Commands

```
SENSe:MIXer:FREQuency:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:STARt
value: float = driver.sense.mixer.frequency.start.get()
```

This command sets or queries the frequency at which the external mixer band starts.

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:STARt
driver.sense.mixer.frequency.start.set(frequency = 1.0)
```

This command sets or queries the frequency at which the external mixer band starts.

param frequency

No help available

6.17.12.2.3 Stop

SCPI Commands

```
SENSe:MIXer:FREQuency:STOP
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:FREQuency:STOP
value: float = driver.sense.mixer.frequency.stop.get()
```

This command sets or queries the frequency at which the external mixer band stops.

return

frequency: No help available

set(frequency: float) → None

```
# SCPI: [SENSe]:MIXer:FREQuency:STOP
driver.sense.mixer.frequency.stop.set(frequency = 1.0)
```

This command sets or queries the frequency at which the external mixer band stops.

param frequency
No help available

6.17.12.3 Harmonic

class HarmonicCls

Harmonic commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.harmonic.clone()
```

Subgroups

6.17.12.3.1 Band

SCPI Commands

```
SENSe:MIXer:HARMonic:BAND
SENSe:MIXer:HARMonic:BAND:PRESet
```

class BandCls

Band commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get() → Band

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
value: enums.Band = driver.sense.mixer.harmonic.band.get()
```

This command selects the external mixer band. The query returns the currently selected band. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]).

return

band: KA | Q | U | V | E | W | F | D | G | Y | J | USER Standard waveguide band or user-defined band.

preset() → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.sense.mixer.harmonic.band.preset()
```

This command restores the preset frequency ranges for the selected standard waveguide band. Note: Changes to the band and mixer settings are maintained even after using the [PRESET] function. Use this command to restore the predefined band ranges.

preset_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND:PRESet
driver.sense.mixer.harmonic.band.preset_with_opc()
```

This command restores the preset frequency ranges for the selected standard waveguide band. Note: Changes to the band and mixer settings are maintained even after using the [PRESET] function. Use this command to restore the predefined band ranges.

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

set(band: Band) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:BAND
driver.sense.mixer.harmonic.band.set(band = enums.Band.A)
```

This command selects the external mixer band. The query returns the currently selected band. This command is only available if the external mixer is active (see [SENSe:]MIXer<x>[:STATe]).

param band

KA|Q|U|V|E|W|F|D|G|Y|J|USER Standard waveguide band or user-defined band.

6.17.12.3.2 High

SCPI Commands

```
SENSe:MIXer:HARMonic:HIGh
```

class HighCls

High commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGh
value: float = driver.sense.mixer.harmonic.high.get()
```

No command help available

return

freq_high: No help available

set(freq_high: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGh
driver.sense.mixer.harmonic.high.set(freq_high = 1.0)
```

No command help available

param freq_high

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.harmonic.high.clone()
```

Subgroups

6.17.12.3.2.1 State

SCPI Commands

```
SENSe:MIXer:HARMonic:HIGH:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
value: bool = driver.sense.mixer.harmonic.high.state.get()
```

This command specifies whether a second (high) harmonic is to be used to cover the band's frequency range.

return

freq_high: No help available

set(freq_high: bool) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:HIGH:STATe
driver.sense.mixer.harmonic.high.state.set(freq_high = False)
```

This command specifies whether a second (high) harmonic is to be used to cover the band's frequency range.

param freq_high

No help available

6.17.12.3.3 Low

SCPI Commands

```
SENSe:MIXer:HARMonic:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
value: float = driver.sense.mixer.harmonic.low.get()
```

This command specifies the harmonic order to be used for the low (first) range.

return

harm_order: Range: 2 to 61 (USER band) ; for other bands: see band definition

set(harm_order: float) → None

```
# SCPI: [SENSe]:MIXer:HARMonic[:LOW]
driver.sense.mixer.harmonic.low.set(harm_order = 1.0)
```

This command specifies the harmonic order to be used for the low (first) range.

param harm_order

Range: 2 to 61 (USER band) ; for other bands: see band definition

6.17.12.3.4 TypePy

SCPI Commands

```
SENSe:MIXer:HARMonic:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → OddEven

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
value: enums.OddEven = driver.sense.mixer.harmonic.typePy.get()
```

This command specifies whether the harmonic order to be used should be odd, even, or both. Which harmonics are supported depends on the mixer type.

return

odd_even: ODD | EVEN | EODD ODD | EVEN | EODD

set(odd_even: OddEven) → None

```
# SCPI: [SENSe]:MIXer:HARMonic:TYPE
driver.sense.mixer.harmonic.typePy.set(odd_even = enums.OddEven.EODD)
```

This command specifies whether the harmonic order to be used should be odd, even, or both. Which harmonics are supported depends on the mixer type.

param odd_even

ODD | EVEN | EODD ODD | EVEN | EODD

6.17.12.4 Ifreq

SCPI Commands

SENSe:MIXer:IF

class IfreqCls

Ifreq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:IF
value: float = driver.sense.mixer.ifreq.get()
```

No command help available

return
frequency: No help available

6.17.12.5 LoPower

SCPI Commands

SENSe:MIXer:LOPower

class LoPowerCls

LoPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOPower
value: float = driver.sense.mixer.loPower.get()
```

This command specifies the LO level of the external mixer's LO port.

return
low_power: No help available

set(low_power: float) → None

```
# SCPI: [SENSe]:MIXer:LOPower
driver.sense.mixer.loPower.set(low_power = 1.0)
```

This command specifies the LO level of the external mixer's LO port.

param low_power
No help available

6.17.12.6 Loss

class LossCls

Loss commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.loss.clone()
```

Subgroups

6.17.12.6.1 High

SCPI Commands

```
SENSe:MIXer:LOSS:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
value: float = driver.sense.mixer.loss.high.get()
```

This command defines the average conversion loss to be used for the entire high (second) range.

return
loss_high: No help available

set(loss_high: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS:HIGH
driver.sense.mixer.loss.high.set(loss_high = 1.0)
```

This command defines the average conversion loss to be used for the entire high (second) range.

param loss_high
No help available

6.17.12.6.2 Low

SCPI Commands

```
SENSe:MIXer:LOSS:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
value: float = driver.sense.mixer.loss.low.get()
```

This command defines the average conversion loss to be used for the entire low (first) range.

return
loss_low: No help available

set(loss_low: float) → None

```
# SCPI: [SENSe]:MIXer:LOSS[:LOW]
driver.sense.mixer.loss.low.set(loss_low = 1.0)
```

This command defines the average conversion loss to be used for the entire low (first) range.

param loss_low
No help available

6.17.12.6.3 Table

class TableCls

Table commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.loss.table.clone()
```

Subgroups

6.17.12.6.3.1 High

SCPI Commands

```
SENSe:MIXer:LOSS:TABLE:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE:HIGH
value: str = driver.sense.mixer.loss.table.high.get()
```

This command defines the conversion loss table to be used for the high (second) range.

return
filename: String containing the path and name of the file, or the serial number of the external mixer whose file is required. The R&S FSWP automatically selects the correct cvl file for the current IF. As an alternative, you can also select a user-defined conversion loss table (.acl file) .

set(filename: str) → None

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE:HIGH
driver.sense.mixer.loss.table.high.set(filename = '1')
```

This command defines the conversion loss table to be used for the high (second) range.

param filename

String containing the path and name of the file, or the serial number of the external mixer whose file is required. The R&S FSWP automatically selects the correct cvl file for the current IF. As an alternative, you can also select a user-defined conversion loss table (.acl file) .

6.17.12.6.3.2 Low

SCPI Commands

```
SENSe:MIXer:LOSS:TABLE:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
value: str = driver.sense.mixer.loss.table.low.get()
```

This command defines the file name of the conversion loss table to be used for the low (first) range.

return

filename: String containing the path and name of the file, or the serial number of the external mixer whose file is required. The R&S FSWP automatically selects the correct cvl file for the current IF. As an alternative, you can also select a user-defined conversion loss table (.acl file) .

set(filename: str) → None

```
# SCPI: [SENSe]:MIXer:LOSS:TABLE[:LOW]
driver.sense.mixer.loss.table.low.set(filename = '1')
```

This command defines the file name of the conversion loss table to be used for the low (first) range.

param filename

String containing the path and name of the file, or the serial number of the external mixer whose file is required. The R&S FSWP automatically selects the correct cvl file for the current IF. As an alternative, you can also select a user-defined conversion loss table (.acl file) .

6.17.12.7 Ports

SCPI Commands

`SENSe:MIXer:PORTs`

class PortsCls

Ports commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:MIXer:PORTs
value: int = driver.sense.mixer.ports.get()
```

This command queries the connected mixer type. Currently, only three-port mixers are supported.

return

port_type: 2 | 3 2 Two-port mixer. 3 Three-port mixer.

set(port_type: int) → None

```
# SCPI: [SENSe]:MIXer:PORTs
driver.sense.mixer.ports.set(port_type = 1)
```

This command queries the connected mixer type. Currently, only three-port mixers are supported.

param port_type

2 | 3 2 Two-port mixer. 3 Three-port mixer.

6.17.12.8 RfOverrange

class RfOverrangeCls

RfOverrange commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mixer.rfOverrange.clone()
```

Subgroups

6.17.12.8.1 State

SCPI Commands

`SENSe:MIXer:RFOVerrange:STATE`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer:RFOVerrange[:STATe]
value: bool = driver.sense.mixer.rf0verrange.state.get()
```

If enabled, the band limits are extended beyond ‘RF Start’ and ‘RF Stop’ due to the capabilities of the used harmonics.

```
return
    rf_overrange_state: No help available
```

set(rf_overrange_state: bool) → None

```
# SCPI: [SENSe]:MIXer:RFOVerrange[:STATe]
driver.sense.mixer.rf0verrange.state.set(rf_overrange_state = False)
```

If enabled, the band limits are extended beyond ‘RF Start’ and ‘RF Stop’ due to the capabilities of the used harmonics.

```
param rf_overrange_state
    No help available
```

6.17.12.9 Signal

SCPI Commands

SENSe:MIXer:SIGNal

class SignalCls

Signal commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → State

```
# SCPI: [SENSe]:MIXer:SIGNal
value: enums.State = driver.sense.mixer.signal.get()
```

No command help available

```
return
    state: No help available
```

set(state: State) → None

```
# SCPI: [SENSe]:MIXer:SIGNal
driver.sense.mixer.signal.set(state = enums.State.ALL)
```

No command help available

```
param state
    No help available
```

6.17.12.10 State

SCPI Commands

SENSe:MIXer:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:MIXer[:STATe]
value: bool = driver.sense.mixer.state.get()
```

Activates or deactivates the use of a connected external mixer as input for the measurement. This command is only available if the optional External Mixer is installed and an external mixer is connected.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: [SENSe]:MIXer[:STATe]
driver.sense.mixer.state.set(state = False)
```

Activates or deactivates the use of a connected external mixer as input for the measurement. This command is only available if the optional External Mixer is installed and an external mixer is connected.

param state

ON | OFF | 1 | 0

6.17.12.11 Threshold

SCPI Commands

SENSe:MIXer:THReshold

class ThresholdCls

Threshold commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MIXer:THReshold
value: float = driver.sense.mixer.threshold.get()
```

No command help available

return

threshold: No help available

set(threshold: float) → None

```
# SCPI: [SENSe]:MIXer:THReshold
driver.sense.mixer.threshold.set(threshold = 1.0)
```

No command help available

param threshold
No help available

6.17.13 Mpower

class MpowerCls

Mpower commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mpower.clone()
```

Subgroups

6.17.13.1 Ftype

SCPI Commands

```
SENSe:MPower:FTYPE
```

class FtypeCls

Ftype commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → FilterTypeC

```
# SCPI: [SENSe]:MPower:FTYPE
value: enums.FilterTypeC = driver.sense.mpower.ftype.get()
```

No command help available

return
filter_type: No help available

set(filter_type: FilterTypeC) → None

```
# SCPI: [SENSe]:MPower:FTYPE
driver.sense.mpower.ftype.set(filter_type = enums.FilterTypeC.CFILTER)
```

No command help available

param filter_type
No help available

6.17.13.2 Result

class ResultCls

Result commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.mpower.result.clone()
```

Subgroups

6.17.13.2.1 ListPy

SCPI Commands

```
SENSe:MPower:RESult:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MPower:RESult[:LIST]
value: float = driver.sense.mpower.result.listPy.get()
```

No command help available

```
return
    pulse_power: No help available
```

6.17.13.2.2 Min

SCPI Commands

```
SENSe:MPower:RESult:MIN
```

class MinCls

Min commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MPower:RESult:MIN
value: float = driver.sense.mpower.result.min.get()
```

No command help available

```
return
    pulse_power: No help available
```

6.17.13.3 Sequence

SCPI Commands

SENSe:MPower:SEquence

class SequenceCls

Sequence commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetStruct

Structure for setting input parameters. Fields:

- Frequency: float: No parameter help available
- Rbw: float: No parameter help available
- Meas_Time: float: No parameter help available
- Trigger_Source: enums.TriggerSourceMpower: No parameter help available
- Trigger_Level: float: No parameter help available
- Trigger_Offset: float: No parameter help available
- Detector: enums.MpowerDetector: No parameter help available
- Of_Pulses: float: No parameter help available

get() → List[float]

```
# SCPI: [SENSe]:MPower[:SEquence]
value: List[float] = driver.sense.mpower.sequence.get()
```

No command help available

return
power_levels: No help available

set(structure: SetStruct) → None

```
# SCPI: [SENSe]:MPower[:SEquence]
structure = driver.sense.mpower.sequence.SetStruct()
structure.Frequency: float = 1.0
structure.Rbw: float = 1.0
structure.Meas_Time: float = 1.0
structure.Trigger_Source: enums.TriggerSourceMpower = enums.TriggerSourceMpower.
↳EXT2
structure.Trigger_Level: float = 1.0
structure.Trigger_Offset: float = 1.0
structure.Detector: enums.MpowerDetector = enums.MpowerDetector.MEAN
structure.Of_Pulses: float = 1.0
driver.sense.mpower.sequence.set(structure)
```

No command help available

param structure
for set value, see the help for SetStruct structure arguments.

6.17.14 Msra

class MsraCls

Msra commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.msra.clone()
```

Subgroups

6.17.14.1 Capture

class CaptureCls

Capture commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.msra.capture.clone()
```

Subgroups

6.17.14.1.1 Offset

SCPI Commands

```
SENSe:MSRA:CAPTure:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:MSRA:CAPTure:OFFSet
value: float = driver.sense.msra.capture.offset.get()
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

return

offset: This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

set(*offset: float*) → None

```
# SCPI: [SENSe]:MSRA:CAPture:OFFSet
driver.sense.msra.capture.offset.set(offset = 1.0)
```

This setting is only available for secondary applications in MSRA mode, not for the MSRA primary application. It has a similar effect as the trigger offset in other measurements.

param offset

This parameter defines the time offset between the capture buffer start and the start of the extracted secondary application data. The offset must be a positive value, as the secondary application can only analyze data that is contained in the capture buffer.
Range: 0 to Record length, Unit: S

6.17.15 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.sense.pmeter.repcap_powerMeter_get()
driver.sense.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 17 total commands, 8 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.clone()
```

Subgroups

6.17.15.1 Dcycle

class DcycleCls

Dcycle commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.dcycle.clone()
```

Subgroups

6.17.15.1.1 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:DCYCLe:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe[:STATe]
value: bool = driver.sense.pmeter.dcycle.state.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(*state: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe[:STATe]
driver.sense.pmeter.dcycle.state.set(state = False, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.1.2 Value

SCPI Commands

```
SENSe:PMETer<PowerMeter>:DCYCLe:VALue
```

class ValueCls

Value commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:DCYCLe:VALue
value: float = driver.sense.pmeter.dcycle.value.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

percentage: No help available

set(percentage: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETER<p>:DCYCLE:VALue
driver.sense.pmeter.dcycle.value.set(percentage = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param percentage

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.2 Frequency

SCPI Commands

SENSe:PMETER<PowerMeter>:FREQUENCY

class FrequencyCls

Frequency commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → float

```
# SCPI: [SENSe]:PMETER<p>:FREQUENCY
value: float = driver.sense.pmeter.frequency.get(powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

frequency: No help available

set(frequency: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETER<p>:FREQUENCY
driver.sense.pmeter.frequency.set(frequency = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param frequency

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.frequency.clone()
```

Subgroups**6.17.15.2.1 Link****SCPI Commands**

```
SENSe:PMETer<PowerMeter>:FREQuency:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → PmeterFreqLink

```
# SCPI: [SENSe]:PMETer<p>:FREQuency:LINK
value: enums.PmeterFreqLink = driver.sense.pmeter.frequency.link.get(powerMeter,
↳= repcap.PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

coupling: No help available

set(coupling: PmeterFreqLink, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:FREQuency:LINK
driver.sense.pmeter.frequency.link.set(coupling = enums.PmeterFreqLink.CENTER,
↳powerMeter = repcap.PowerMeter.Default)
```

No command help available

param coupling

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.3 Mtime

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME
```

class MtimeCls

Mtime commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → Duration

```
# SCPI: [SENSe]:PMETer<p>:MTIME
value: enums.Duration = driver.sense.pmeter.mtime.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

duration: No help available

set(*duration: Duration, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME
driver.sense.pmeter.mtime.set(duration = enums.Duration.LONG, powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param duration

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.mtime.clone()
```

Subgroups

6.17.15.3.1 Average

class AverageCls

Average commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.mtime.average.clone()
```

Subgroups

6.17.15.3.1.1 Count

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME:AVERage:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage:COUNT
value: float = driver.sense.pmeter.mtime.average.count.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

number_readings: No help available

set(*number_readings: float, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage:COUNT
driver.sense.pmeter.mtime.average.count.set(number_readings = 1.0, powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param number_readings

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.3.1.2 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:MTIME:AVERage:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage[:STATe]
value: bool = driver.sense.pmeter.mtime.average.state.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(*state: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:MTIME:AVERage[:STATe]
driver.sense.pmeter.mtime.average.state.set(state = False, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.4 Roffset

class RoffsetCls

Roffset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.roffset.clone()
```

Subgroups

6.17.15.4.1 State

SCPI Commands

SENSe:PMETer<PowerMeter>:ROFFset:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:ROFFset[:STATe]
value: bool = driver.sense.pmeter.roffset.state.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(*state: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:ROFFset[:STATe]
driver.sense.pmeter.roffset.state.set(state = False, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.5 Soffset

SCPI Commands

SENSe:PMETer<PowerMeter>:SOFFset

class SoffsetCls

Soffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:SOFFset
value: float = driver.sense.pmeter.soffset.get(powerMeter = repcap.PowerMeter.
↳Default)
```


No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

sensor_offset: No help available

set(sensor_offset: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETER<p>:SOFFset
driver.sense.pmeter.soffset.set(sensor_offset = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param sensor_offset

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.6 State

SCPI Commands

SENSe:PMETER<PowerMeter>:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → bool

```
# SCPI: [SENSe]:PMETER<p>[:STATe]
value: bool = driver.sense.pmeter.state.get(powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(state: bool, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETER<p>[:STATe]
driver.sense.pmeter.state.set(state = False, powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7 Trigger**class TriggerCls**

Trigger commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.trigger.clone()
```

Subgroups**6.17.15.7.1 Dtime****SCPI Commands**

```
SENSe:PMETer<PowerMeter>:TRIGger:DTIME
```

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → float

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:DTIME
value: float = driver.sense.pmeter.trigger.dtime.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

time: No help available

set(time: float, powerMeter=PowerMeter.Default) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:DTIME
driver.sense.pmeter.trigger.dtime.set(time = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param time

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7.2 Holdoff**SCPI Commands**

```
SENSe:PMETer<PowerMeter>:TRIGger:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:HOLDoff
value: float = driver.sense.pmeter.trigger.holdoff.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

holdoff: No help available

set(*holdoff: float, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:HOLDoff
driver.sense.pmeter.trigger.holdoff.set(holdoff = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param holdoff

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7.3 Hysteresis**SCPI Commands**

```
SENSe:PMETer<PowerMeter>:TRIGger:HYSTeresis
```

class HysteresisCls

Hysteresis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETER<p>:TRIGger:HYSTeresis
value: float = driver.sense.pmeter.trigger.hysteresis.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

hysteresis: No help available

set(*hysteresis: float, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETER<p>:TRIGger:HYSTeresis
driver.sense.pmeter.trigger.hysteresis.set(hysteresis = 1.0, powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param hysteresis

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7.4 Level

SCPI Commands

```
SENSe:PMETER<PowerMeter>:TRIGger:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: [SENSe]:PMETER<p>:TRIGger:LEVel
value: float = driver.sense.pmeter.trigger.level.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

level: No help available

set(*level: float, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:LEVel
driver.sense.pmeter.trigger.level.set(level = 1.0, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param level

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7.5 Slope

SCPI Commands

```
SENSe:PMETer<PowerMeter>:TRIGger:SLOPe
```

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → SlopeType

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:SLOPe
value: enums.SlopeType = driver.sense.pmeter.trigger.slope.get(powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

edge: No help available

set(*edge: SlopeType, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger:SLOPe
driver.sense.pmeter.trigger.slope.set(edge = enums.SlopeType.NEGative,
↳powerMeter = repcap.PowerMeter.Default)
```

No command help available

param edge

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.7.6 State

SCPI Commands

`SENSe:PMETer<PowerMeter>:TRIGger:STATe`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:TRIGger[:STATe]
value: bool = driver.sense.pmeter.trigger.state.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(*state: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:TRIGger[:STATe]
driver.sense.pmeter.trigger.state.set(state = False, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.15.8 Update

class UpdateCls

Update commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.pmeter.update.clone()
```

Subgroups

6.17.15.8.1 State

SCPI Commands

```
SENSe:PMETer<PowerMeter>:UPDate:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → bool

```
# SCPI: [SENSe]:PMETer<p>:UPDate[:STATe]
value: bool = driver.sense.pmeter.update.state.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(*state: bool, powerMeter=PowerMeter.Default*) → None

```
# SCPI: [SENSe]:PMETer<p>:UPDate[:STATe]
driver.sense.pmeter.update.state.set(state = False, powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.17.16 Power

class PowerCls

Power commands group definition. 66 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.clone()
```

Subgroups

6.17.16.1 Achannel

SCPI Commands

```
SENSe:POWer:ACHannel:PRESet
```

class AchannelCls

Achannel commands group definition. 62 total commands, 15 Subgroups, 1 group commands

preset(*measurement: PowerMeasFunction*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:PRESet
driver.sense.power.achannel.preset(measurement = enums.PowerMeasFunction.
↳ ACPower)
```

No command help available

param measurement

No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.clone()
```

Subgroups

6.17.16.1.1 AcPairs

SCPI Commands

```
SENSe:POWer:ACHannel:ACPairs
```

class AcPairsCls

AcPairs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:ACPairs
value: float = driver.sense.power.achannel.acPairs.get()
```

No command help available

return
 channel_pairs: No help available
set(channel_pairs: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:ACPairs
driver.sense.power.achannel.acPairs.set(channel_pairs = 1.0)
```

No command help available

param channel_pairs
 No help available

6.17.16.1.2 AgChannels

SCPI Commands

```
SENSe:POWer:ACHannel:AGChannels
```

class AgChannelsCls

AgChannels commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:ACHannel:AGChannels
value: bool = driver.sense.power.achannel.agChannels.get()
```

No command help available

return
 state: No help available
set(state: bool) → None

```
# SCPI: [SENSe]:POWer:ACHannel:AGChannels
driver.sense.power.achannel.agChannels.set(state = False)
```

No command help available

param state
 No help available

6.17.16.1.3 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 5 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.bandwidth.clone()
```

Subgroups

6.17.16.1.3.1 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.bandwidth.gap.repcap_gapChannel_get()
driver.sense.power.achannel.bandwidth.gap.repcap_gapChannel_set(repcap.GapChannel.Nr1)
```

class GapCls

Gap commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.bandwidth.gap.clone()
```

Subgroups

6.17.16.1.3.2 Auto

SCPI Commands

```
SENSe:POWer:ACHannel:BWIDth:GAP<GapChannel>:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(bandwidth: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:GAP<gap>[:AUTO]
driver.sense.power.achannel.bandwidth.gap.auto.set(bandwidth = 1.0, gapChannel_
↪ = repcap.GapChannel.Default)
```

No command help available

param bandwidth
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.3.3 Manual

class ManualCls

Manual commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.bandwidth.gap.manual.clone()
```

Subgroups

6.17.16.1.3.4 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:BWIDth:GAP<GapChannel>:MANual:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:GAP<gap>:MANual:LOWer
value: float = driver.sense.power.achannel.bandwidth.gap.manual.lower.get(sb_
↳ gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

bandwidth: No help available

set(sb_gaps: SubBlockGaps, bandwidth: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:GAP<gap>:MANual:LOWer
driver.sense.power.achannel.bandwidth.gap.manual.lower.set(sb_gaps = enums.
↳ SubBlockGaps.AB, bandwidth = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param bandwidth

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.3.5 Upper**SCPI Commands**

`SENSe:POWer:ACHannel:BWIDth:GAP<GapChannel>:MANual:UPPer`

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *gapChannel*=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:GAP<gap>:MANual:UPPer
value: float = driver.sense.power.achannel.bandwidth.gap.manual.upper.get(sb_
↳ gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

bandwidth: No help available

set(*sb_gaps*: SubBlockGaps, *bandwidth*: float, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:GAP<gap>:MANual:UPPer
driver.sense.power.achannel.bandwidth.gap.manual.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, bandwidth = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param bandwidth

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.3.6 UaChannel

SCPI Commands

```
SENSe:POWer:ACHannel:BWIDth:UACHannel
```

class UaChannelCls

UaChannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:UACHannel
value: float = driver.sense.power.achannel.bandwidth.uaChannel.get()
```

No command help available

return

bandwidth: No help available

set(bandwidth: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:UACHannel
driver.sense.power.achannel.bandwidth.uaChannel.set(bandwidth = 1.0)
```

No command help available

param bandwidth

No help available

6.17.16.1.3.7 Ualternate<UpperAltChannel>

RepCap Settings

```
# Range: Nr1 .. Nr63
rc = driver.sense.power.achannel.bandwidth.ualternate.repcap_upperAltChannel_get()
driver.sense.power.achannel.bandwidth.ualternate.repcap_upperAltChannel_set(repcap.
↳UpperAltChannel.Nr1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:BWIDth:UALTernate<UpperAltChannel>
```

class UalternateCls

Ualternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: UpperAltChannel, default value after init: UpperAltChannel.Nr1

get(upperAltChannel=UpperAltChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:UALTernate<ch>
value: float = driver.sense.power.achannel.bandwidth.ualternate.
↳get(upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Ualternate’)

return

bandwidth: No help available

set(bandwidth: float, upperAltChannel=UpperAltChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:BWIDth:UALternate<ch>
driver.sense.power.achannel.bandwidth.ualternate.set(bandwidth = 1.0,
↪upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param bandwidth

No help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Ualternate’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.bandwidth.ualternate.clone()
```

6.17.16.1.4 FilterPy

class FilterPyCls

FilterPy commands group definition. 20 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.clone()
```

Subgroups

6.17.16.1.4.1 Alpha

class AlphaCls

Alpha commands group definition. 10 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.clone()
```

Subgroups

6.17.16.1.4.2 Achannel

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:ACHannel
```

class AchannelCls

Achannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:ACHannel
value: float = driver.sense.power.achannel.filterPy.alpha.achannel.get()
```

No command help available

return

alpha: No help available

set(alpha: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:ACHannel
driver.sense.power.achannel.filterPy.alpha.achannel.set(alpha = 1.0)
```

No command help available

param alpha

No help available

6.17.16.1.4.3 All

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(value: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa[:ALL]
driver.sense.power.achannel.filterPy.alpha.all.set(value = 1.0)
```

No command help available

param value
No help available

6.17.16.1.4.4 Alternate<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.alpha.alternate.repcap_channel_get()
driver.sense.power.achannel.filterPy.alpha.alternate.repcap_channel_set(repcap.Channel.
↪ Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:ALternate<Channel>
```

class AlternateCls

Alternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(*channel=Channel.Default*) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:ALternate<ch>
value: float = driver.sense.power.achannel.filterPy.alpha.alternate.get(channel.
↪ repcap.Channel.Default)
```

No command help available

param channel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return
alpha: No help available

set(*alpha: float, channel=Channel.Default*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:ALternate<ch>
driver.sense.power.achannel.filterPy.alpha.alternate.set(alpha = 1.0, channel =
↪ repcap.Channel.Default)
```

No command help available

param alpha
No help available

param channel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.alternate.clone()
```

6.17.16.1.4.5 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.alpha.channel.repcap_channel_get()
driver.sense.power.achannel.filterPy.alpha.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:CHANnel<ch>
value: float = driver.sense.power.achannel.filterPy.alpha.channel.get(channel = ↵
↵repcap.Channel.Default)
```

No command help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

alpha: No help available

set(alpha: float, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:CHANnel<ch>
driver.sense.power.achannel.filterPy.alpha.channel.set(alpha = 1.0, channel = ↵
↵repcap.Channel.Default)
```

No command help available

param alpha

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.channel.clone()
```

6.17.16.1.4.6 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.filterPy.alpha.gap.repcap_gapChannel_get()
driver.sense.power.achannel.filterPy.alpha.gap.repcap_gapChannel_set(repcap.GapChannel.
↪Nr1)
```

class GapCls

Gap commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.gap.clone()
```

Subgroups

6.17.16.1.4.7 Auto

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:GAP<GapChannel>:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(alpha: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:GAP<gap>[:AUTO]
driver.sense.power.achannel.filterPy.alpha.gap.auto.set(alpha = 1.0, gapChannel_
↪= repcap.GapChannel.Default)
```

No command help available

param alpha

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.8 Manual

class ManualCls

Manual commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.gap.manual.clone()
```

Subgroups

6.17.16.1.4.9 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:GAP<GapChannel>:MANual:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:GAP<gap>:MANual:LOWer
value: float = driver.sense.power.achannel.filterPy.alpha.gap.manual.lower.
↳get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

alpha: No help available

set(sb_gaps: SubBlockGaps, alpha: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:GAP<gap>:MANual:LOWer
driver.sense.power.achannel.filterPy.alpha.gap.manual.lower.set(sb_gaps = enums.
↳SubBlockGaps.AB, alpha = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param alpha

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.10 Upper**SCPI Commands**

`SENSe:POWer:ACHannel:FILTer:ALPHa:GAP<GapChannel>:MANual:UPPer`

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *gapChannel*=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:GAP<gap>:MANual:UPPer
value: float = driver.sense.power.achannel.filterPy.alpha.gap.manual.upper.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

alpha: No help available

set(*sb_gaps*: SubBlockGaps, *alpha*: float, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:GAP<gap>:MANual:UPPer
driver.sense.power.achannel.filterPy.alpha.gap.manual.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, alpha = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param alpha

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.11 Sblock<SubBlock>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.sense.power.achannel.filterPy.alpha.sblock.repcap_subBlock_get()
driver.sense.power.achannel.filterPy.alpha.sblock.repcap_subBlock_set(repcap.SubBlock.
↳Nr1)
```

class SblockCls

Sblock commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.sblock.clone()
```

Subgroups

6.17.16.1.4.12 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.alpha.sblock.channel.repcap_channel_get()
driver.sense.power.achannel.filterPy.alpha.sblock.channel.repcap_channel_set(repcap.
↳Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHA:SBlock<SubBlock>:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHA:SBlock<sb>:CHANnel<ch>
value: float = driver.sense.power.achannel.filterPy.alpha.sblock.channel.
↳get(subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

alpha: No help available

set(*alpha: float, subBlock=SubBlock.Default, channel=Channel.Default*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FiLter:ALPHA:SBLock<sb>:CHANnel<ch>
driver.sense.power.achannel.filterPy.alpha.sblock.channel.set(alpha = 1.0,
↳subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param alpha

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.sblock.channel.clone()
```

6.17.16.1.4.13 UaChannel**SCPI Commands**

```
SENSe:POWer:ACHannel:FiLter:ALPHA:UACHannel
```

class UaChannelCls

UaChannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:FiLter:ALPHA:UACHannel
value: float = driver.sense.power.achannel.filterPy.alpha.uaChannel.get()
```

No command help available

return

alpha: No help available

set(*alpha: float*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FiLter:ALPHA:UACHannel
driver.sense.power.achannel.filterPy.alpha.uaChannel.set(alpha = 1.0)
```

No command help available

param alpha

No help available

6.17.16.1.4.14 Ualternate<UpperAltChannel>

RepCap Settings

```
# Range: Nr1 .. Nr63
rc = driver.sense.power.achannel.filterPy.alpha.ualternate.repcap_upperAltChannel_get()
driver.sense.power.achannel.filterPy.alpha.ualternate.repcap_upperAltChannel_set(repcap.
↳UpperAltChannel.Nr1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:ALPHa:UALTernate<UpperAltChannel>
```

class UalternateCls

Ualternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: UpperAltChannel, default value after init: UpperAltChannel.Nr1

get(upperAltChannel=UpperAltChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:UALTernate<ch>
value: float = driver.sense.power.achannel.filterPy.alpha.ualternate.
↳get(upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

return

alpha: No help available

set(alpha: float, upperAltChannel=UpperAltChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer:ALPHa:UALTernate<ch>
driver.sense.power.achannel.filterPy.alpha.ualternate.set(alpha = 1.0,
↳upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param alpha

No help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.alpha.ualternate.clone()
```

6.17.16.1.4.15 State

class StateCls

State commands group definition. 10 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.clone()
```

Subgroups

6.17.16.1.4.16 Achannel

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:ACHannel
```

class AchannelCls

Achannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:ACHannel
value: bool = driver.sense.power.achannel.filterPy.state.achannel.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:ACHannel
driver.sense.power.achannel.filterPy.state.achannel.set(state = False)
```

No command help available

param state

No help available

6.17.16.1.4.17 All

SCPI Commands

```
SENSe:POWer:ACHannel:FILTeR:STaTe:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*state: bool*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTeR[:STaTe][:ALL]
driver.sense.power.achannel.filterPy.state.all.set(state = False)
```

No command help available

param state

No help available

6.17.16.1.4.18 Alternate<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.state.alternate.repcap_channel_get()
driver.sense.power.achannel.filterPy.state.alternate.repcap_channel_set(repcap.Channel.
↪ Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTeR:STaTe:ALTeRnate<Channel>
```

class AlternateCls

Alternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(*channel=Channel.Default*) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTeR[:STaTe]:ALTeRnate<ch>
value: bool = driver.sense.power.achannel.filterPy.state.alternate.get(channel.
↪ repcap.Channel.Default)
```

No command help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return

state: No help available

set(state: bool, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:ALternate<ch>
driver.sense.power.achannel.filterPy.state.alternate.set(state = False, channel_
↪= repcap.Channel.Default)
```

No command help available

param state

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.alternate.clone()
```

6.17.16.1.4.19 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.state.channel.repcap_channel_get()
driver.sense.power.achannel.filterPy.state.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:CHANnel<ch>
value: bool = driver.sense.power.achannel.filterPy.state.channel.get(channel =
↪repcap.Channel.Default)
```

No command help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

state: No help available

set(state: bool, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FiLTeR[:STATe]:CHANnel<ch>
driver.sense.power.achannel.filterPy.state.channel.set(state = False, channel =
↳repcap.Channel.Default)
```

No command help available

param state

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.channel.clone()
```

6.17.16.1.4.20 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.filterPy.state.gap.repcap_gapChannel_get()
driver.sense.power.achannel.filterPy.state.gap.repcap_gapChannel_set(repcap.GapChannel.
↳Nr1)
```

class GapCls

Gap commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.gap.clone()
```

Subgroups

6.17.16.1.4.21 Auto

SCPI Commands

```
SENSe:POWer:ACHannel:FiLTeR:STATe:GAP<GapChannel>:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:GAP<gap>[:AUTO]
driver.sense.power.achannel.filterPy.state.gap.auto.set(state = False,
↳ gapChannel = repcap.GapChannel.Default)
```

No command help available

param state

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.22 Manual

class ManualCls

Manual commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.gap.manual.clone()
```

Subgroups

6.17.16.1.4.23 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:GAP<GapChannel>:MANual:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:GAP<gap>:MANual:LOWer
value: bool = driver.sense.power.achannel.filterPy.state.gap.manual.lower.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(*sb_gaps*: SubBlockGaps, *state*: bool, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:GAP<gap>:MANual:LOWer
driver.sense.power.achannel.filterPy.state.gap.manual.lower.set(sb_gaps = enums.
↳ SubBlockGaps.AB, state = False, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param state
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.24 Upper

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:GAP<GapChannel>:MANual:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *gapChannel*=GapChannel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:GAP<gap>:MANual:UPPer
value: bool = driver.sense.power.achannel.filterPy.state.gap.manual.upper.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return
state: No help available

set(*sb_gaps*: SubBlockGaps, *state*: bool, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:GAP<gap>:MANual:UPPer
driver.sense.power.achannel.filterPy.state.gap.manual.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, state = False, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param state

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.4.25 Sblock<SubBlock>**RepCap Settings**

```
# Range: Nr1 .. Nr8
rc = driver.sense.power.achannel.filterPy.state.sblock.repcap_subBlock_get()
driver.sense.power.achannel.filterPy.state.sblock.repcap_subBlock_set(repcap.SubBlock.
↳Nr1)
```

class SblockCls

Sblock commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.sblock.clone()
```

Subgroups**6.17.16.1.4.26 Channel<Channel>****RepCap Settings**

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.filterPy.state.sblock.channel.repcap_channel_get()
driver.sense.power.achannel.filterPy.state.sblock.channel.repcap_channel_set(repcap.
↳Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:SBLOCK<SubBlock>:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:SBLOCK<sb>:CHANnel<ch>
value: bool = driver.sense.power.achannel.filterPy.state.sblock.channel.
↳get(subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

state: No help available

set(state: bool, subBlock=SubBlock.Default, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FiLTeR[:STATe]:SBLoCk<sb>:CHANnel<ch>
driver.sense.power.achannel.filterPy.state.sblock.channel.set(state = False,
↳ subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param state

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.sblock.channel.clone()
```

6.17.16.1.4.27 UaChannel

SCPI Commands

```
SENSe:POWer:ACHannel:FiLTeR:STATe:UAChannel
```

class UaChannelCls

UaChannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FiLTeR[:STATe]:UAChannel
value: bool = driver.sense.power.achannel.filterPy.state.uaChannel.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:UACHannel
driver.sense.power.achannel.filterPy.state.uaChannel.set(state = False)
```

No command help available

param state

No help available

6.17.16.1.4.28 Ualternate<UpperAltChannel>

RepCap Settings

```
# Range: Nr1 .. Nr63
rc = driver.sense.power.achannel.filterPy.state.ualternate.repcap_upperAltChannel_get()
driver.sense.power.achannel.filterPy.state.ualternate.repcap_upperAltChannel_set(repcap.
↪UpperAltChannel.Nr1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:FILTer:STATe:UALTernate<UpperAltChannel>
```

class UalternateCls

Ualternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: UpperAltChannel, default value after init: UpperAltChannel.Nr1

get(upperAltChannel=UpperAltChannel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:UALTernate<ch>
value: bool = driver.sense.power.achannel.filterPy.state.ualternate.
↪get(upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ual-ternate')

return

state: No help available

set(state: bool, upperAltChannel=UpperAltChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:FILTer[:STATe]:UALTernate<ch>
driver.sense.power.achannel.filterPy.state.ualternate.set(state = False,
↪upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param state

No help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.filterPy.state.ualternate.clone()
```

6.17.16.1.5 Gap<GapChannel>**RepCap Settings**

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.gap.repcap_gapChannel_get()
driver.sense.power.achannel.gap.repcap_gapChannel_set(repcap.GapChannel.Nr1)
```

class GapCls

Gap commands group definition. 4 total commands, 3 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gap.clone()
```

Subgroups**6.17.16.1.5.1 Auto****class AutoCls**

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gap.auto.clone()
```

Subgroups

6.17.16.1.5.2 Msize

SCPI Commands

`SENSe:POWer:ACHannel:GAP<GapChannel>:AUTO:MSIZE`

class MsizeCls

Msize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>[:AUTO]:MSIZE
value: float = driver.sense.power.achannel.gap.auto.msize.get(gapChannel = ↵
↵repcap.GapChannel.Default)
```

No command help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

bandwidth: No help available

set(bandwidth: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>[:AUTO]:MSIZE
driver.sense.power.achannel.gap.auto.msize.set(bandwidth = 1.0, gapChannel = ↵
↵repcap.GapChannel.Default)
```

No command help available

param bandwidth

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.5.3 Manual

class ManualCls

Manual commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gap.manual.clone()
```

Subgroups

6.17.16.1.5.4 Channel

class ChannelCls

Channel commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gap.manual.channel.clone()
```

Subgroups

6.17.16.1.5.5 Count

class CountCls

Count commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gap.manual.channel.count.clone()
```

Subgroups

6.17.16.1.5.6 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:GAP<GapChannel>:MANual:CHANnel:COUNT:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MANual:CHANnel:COUNT:LOWer
value: float = driver.sense.power.achannel.gap.manual.channel.count.lower.
    get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

count: No help available

set(sb_gaps: SubBlockGaps, count: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MANual:CHANnel:COUNT:LOWer
driver.sense.power.achannel.gap.manual.channel.count.lower.set(sb_gaps = enums.
↳ SubBlockGaps.AB, count = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param count

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.5.7 Upper

SCPI Commands

```
SENSe:POWer:ACHannel:GAP<GapChannel>:MANual:CHANnel:COUNT:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MANual:CHANnel:COUNT:UPPer
value: float = driver.sense.power.achannel.gap.manual.channel.count.upper.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

count: No help available

set(*sb_gaps*: SubBlockGaps, *count*: float, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MANual:CHANnel:COUnT:UPPer
driver.sense.power.achannel.gap.manual.channel.count.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, count = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param count
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.5.8 Mode

SCPI Commands

```
SENSe:POWer:ACHannel:GAP<GapChannel>:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*gapChannel*=GapChannel.Default) → AutoManualMode

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MODE
value: enums.AutoManualMode = driver.sense.power.achannel.gap.mode.
↳ get(gapChannel = repcap.GapChannel.Default)
```

No command help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return
mode: No help available

set(*mode*: AutoManualMode, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GAP<gap>:MODE
driver.sense.power.achannel.gap.mode.set(mode = enums.AutoManualMode.AUTO,
↳ gapChannel = repcap.GapChannel.Default)
```

No command help available

param mode
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.6 Gchannel

class GchannelCls

Gchannel commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gchannel.clone()
```

Subgroups

6.17.16.1.6.1 State

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gchannel.state.clone()
```

Subgroups

6.17.16.1.6.2 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.gchannel.state.gap.repcap_gapChannel_get()
driver.sense.power.achannel.gchannel.state.gap.repcap_gapChannel_set(repcap.GapChannel.
↪Nr1)
```

class GapCls

Gap commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gchannel.state.gap.clone()
```

Subgroups

6.17.16.1.6.3 Manual

class ManualCls

Manual commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.gchannel.state.gap.manual.clone()
```

Subgroups

6.17.16.1.6.4 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:GCHannel:STATe:GAP<GapChannel>:MANual:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:GCHannel[:STATe]:GAP<gap>:MANual:LOWer
value: bool = driver.sense.power.achannel.gchannel.state.gap.manual.lower.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(sb_gaps: SubBlockGaps, state: bool, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GCHannel[:STATe]:GAP<gap>:MANual:LOWer
driver.sense.power.achannel.gchannel.state.gap.manual.lower.set(sb_gaps = enums.
↳ SubBlockGaps.AB, state = False, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.6.5 Upper**SCPI Commands**

`SENSe:POWer:ACHannel:GCHannel:STATe:GAP<GapChannel>:MANual:UPPer`

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *gapChannel*=GapChannel.Default) → bool

```
# SCPI: [SENSe]:POWer:ACHannel:GCHannel[:STATe]:GAP<gap>:MANual:UPPer
value: bool = driver.sense.power.achannel.gchannel.state.gap.manual.upper.
↳ get(sb_gaps = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

state: No help available

set(*sb_gaps*: SubBlockGaps, *state*: bool, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:GCHannel[:STATe]:GAP<gap>:MANual:UPPer
driver.sense.power.achannel.gchannel.state.gap.manual.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, state = False, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param state

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.7 Mode

SCPI Commands

```
SENSe:POWer:ACHannel:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ReferenceMode

```
# SCPI: [SENSe]:POWer:ACHannel:MODE
value: enums.ReferenceMode = driver.sense.power.achannel.mode.get()
```

No command help available

return
mode: No help available

set(mode: ReferenceMode) → None

```
# SCPI: [SENSe]:POWer:ACHannel:MODE
driver.sense.power.achannel.mode.set(mode = enums.ReferenceMode.Absolute)
```

No command help available

param mode
No help available

6.17.16.1.8 Name

class NameCls

Name commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.name.clone()
```

Subgroups

6.17.16.1.8.1 Achannel

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:ACHannel
```

class AchannelCls

Achannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:ACHannel
value: str = driver.sense.power.achannel.name.achannel.get()
```

No command help available

return
name: No help available

set(name: str) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:ACHannel
driver.sense.power.achannel.name.achannel.set(name = '1')
```

No command help available

param name
No help available

6.17.16.1.8.2 Alternate<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.name.alternate.repcap_channel_get()
driver.sense.power.achannel.name.alternate.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:ALternate<Channel>
```

class AlternateCls

Alternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:ALternate<ch>
value: str = driver.sense.power.achannel.name.alternate.get(channel = repcap.
↪Channel.Default)
```

No command help available

param channel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

return
name: No help available

set(name: str, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:ALternate<ch>
driver.sense.power.achannel.name.alternate.set(name = '1', channel = repcap.
↳Channel.Default)
```

No command help available

param name

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.name.alternate.clone()
```

6.17.16.1.8.3 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.name.channel.repcap_channel_get()
driver.sense.power.achannel.name.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:CHANnel<ch>
value: str = driver.sense.power.achannel.name.channel.get(channel = repcap.
↳Channel.Default)
```

No command help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

name: No help available

set(name: str, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:CHANnel<ch>
driver.sense.power.achannel.name.channel.set(name = '1', channel = repcap.
↳ Channel.Default)
```

No command help available

param name

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.name.channel.clone()
```

6.17.16.1.8.4 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.name.gap.repcap_gapChannel_get()
driver.sense.power.achannel.name.gap.repcap_gapChannel_set(repcap.GapChannel.Nr1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:GAP<GapChannel>
```

class GapCls

Gap commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

get(gapChannel=GapChannel.Default) → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:GAP<gap>
value: str = driver.sense.power.achannel.name.gap.get(gapChannel = repcap.
↳ GapChannel.Default)
```

No command help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

name: No help available

set(name: str, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:GAP<gap>
driver.sense.power.achannel.name.gap.set(name = '1', gapChannel = repcap.
↳ GapChannel.Default)
```

No command help available

param name

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.name.gap.clone()
```

6.17.16.1.8.5 UaChannel

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:UACHannel
```

class UaChannelCls

UaChannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:UACHannel
value: str = driver.sense.power.achannel.name.uaChannel.get()
```

No command help available

return

name: No help available

set(name: str) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:UACHannel
driver.sense.power.achannel.name.uaChannel.set(name = '1')
```

No command help available

param name

No help available

6.17.16.1.8.6 Ualternate<UpperAltChannel>

RepCap Settings

```
# Range: Nr1 .. Nr63
rc = driver.sense.power.achannel.name.ualternate.repcap_upperAltChannel_get()
driver.sense.power.achannel.name.ualternate.repcap_upperAltChannel_set(repcap.
↳ UpperAltChannel.Nr1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:NAME:UALTernate<UpperAltChannel>
```

class UalternateCls

Ualternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: UpperAltChannel, default value after init: UpperAltChannel.Nr1

get(upperAltChannel=UpperAltChannel.Default) → str

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:UALTernate<ch>
value: str = driver.sense.power.achannel.name.ualternate.get(upperAltChannel =
↳ repcap.UpperAltChannel.Default)
```

No command help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

return

name: No help available

set(name: str, upperAltChannel=UpperAltChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:NAME:UALTernate<ch>
driver.sense.power.achannel.name.ualternate.set(name = '1', upperAltChannel =
↳ repcap.UpperAltChannel.Default)
```

No command help available

param name

No help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.name.ualternate.clone()
```

6.17.16.1.9 PresetRefLevel

SCPI Commands

```
SENSe:POWer:ACHannel:PRESet:RLEVel
```

class PresetRefLevelCls

PresetRefLevel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(opc_timeout_ms: int = -1) → None

```
# SCPI: [SENSe]:POWer:ACHannel:PRESet:RLEVel
driver.sense.power.achannel.presetRefLevel.set()
```

No command help available

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.17.16.1.10 Reference

class ReferenceCls

Reference commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.reference.clone()
```

Subgroups

6.17.16.1.10.1 TxChannel

class TxChannelCls

TxChannel commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.reference.txChannel.clone()
```

Subgroups

6.17.16.1.10.2 Auto

SCPI Commands

```
SENSe:POWer:ACHannel:REFeRence:TXCHannel:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(ref_channel: RefChannel) → None

```
# SCPI: [SENSe]:POWer:ACHannel:REFeRence:TXCHannel:AUTO
driver.sense.power.achannel.reference.txChannel.auto.set(ref_channel = enums.
↳RefChannel.LHIGhest)
```

No command help available

param ref_channel
No help available

6.17.16.1.10.3 Manual

SCPI Commands

```
SENSe:POWer:ACHannel:REFeRence:TXCHannel:MANual
```

class ManualCls

Manual commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:REFeRence:TXCHannel:MANual
value: float = driver.sense.power.achannel.reference.txChannel.manual.get()
```

No command help available

return
channel_number: No help available

set(channel_number: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:REFeRence:TXCHannel:MANual
driver.sense.power.achannel.reference.txChannel.manual.set(channel_number = 1.0)
```

No command help available

param channel_number

No help available

6.17.16.1.11 Sbcount

SCPI Commands

```
SENSe:POWer:ACHannel:SBCount
```

class SbcountCls

Sbcount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBCount
value: float = driver.sense.power.achannel.sbcount.get()
```

No command help available

return

number: No help available

set(number: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBCount
driver.sense.power.achannel.sbcount.set(number = 1.0)
```

No command help available

param number

No help available

6.17.16.1.12 Sblock<SubBlock>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.sense.power.achannel.sblock.repcap_subBlock_get()
driver.sense.power.achannel.sblock.repcap_subBlock_set(repcap.SubBlock.Nr1)
```

class SblockCls

Sblock commands group definition. 7 total commands, 7 Subgroups, 0 group commands Repeated Capability: SubBlock, default value after init: SubBlock.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.clone()
```

Subgroups

6.17.16.1.12.1 Bandwidth

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.bandwidth.clone()
```

Subgroups

6.17.16.1.12.2 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.sblock.bandwidth.channel.repcap_channel_get()
driver.sense.power.achannel.sblock.bandwidth.channel.repcap_channel_set(repcap.Channel.
↪Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SBlock<SubBlock>:BWIDth:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:BWIDth[:CHANnel<ch>]
value: float = driver.sense.power.achannel.sblock.bandwidth.channel.
↪get(subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

bandwidth: No help available

set(bandwidth: float, subBlock=SubBlock.Default, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBLOCK<sb>:BWIDth[:CHANnel<ch>]
driver.sense.power.achannel.sblock.bandwidth.channel.set(bandwidth = 1.0,
↳subBlock = repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param bandwidth

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.bandwidth.channel.clone()
```

6.17.16.1.12.3 Center**class CenterCls**

Center commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.center.clone()
```

Subgroups**6.17.16.1.12.4 Channel<Channel>****RepCap Settings**

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.sblock.center.channel.repcap_channel_get()
driver.sense.power.achannel.sblock.center.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SBlock<SubBlock>:CENTer:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:CENTer[:CHANnel<ch>]
value: float = driver.sense.power.achannel.sblock.center.channel.get(subBlock = ↵
↵repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

frequency: No help available

set(frequency: float, subBlock=SubBlock.Default, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:CENTer[:CHANnel<ch>]
driver.sense.power.achannel.sblock.center.channel.set(frequency = 1.0, subBlock ↵
↵= repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param frequency

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.center.channel.clone()
```

6.17.16.1.12.5 Frequency

class FrequencyCls

Frequency commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.frequency.clone()
```

Subgroups

6.17.16.1.12.6 Center

SCPI Commands

```
SENSe:POWer:ACHannel:SBlock<SubBlock>:FREQuency:CENTer
```

class CenterCls

Center commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:FREQuency:CENTer
value: float = driver.sense.power.achannel.sblock.frequency.center.get(subBlock,
↳ repcap.SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

return

frequency: No help available

set(frequency: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:FREQuency:CENTer
driver.sense.power.achannel.sblock.frequency.center.set(frequency = 1.0,
↳ subBlock = repcap.SubBlock.Default)
```

No command help available

param frequency

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

6.17.16.1.12.7 Name

class NameCls

Name commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.name.clone()
```

Subgroups

6.17.16.1.12.8 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.sblock.name.channel.repcap_channel_get()
driver.sense.power.achannel.sblock.name.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SBLOCK<SubBlock>:NAME:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → str

```
# SCPI: [SENSe]:POWer:ACHannel:SBLOCK<sb>:NAME[:CHANnel<ch>]
value: str = driver.sense.power.achannel.sblock.name.channel.get(subBlock =
↳repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

name: No help available

set(name: str, subBlock=SubBlock.Default, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:NAME[:CHANnel<ch>]
driver.sense.power.achannel.sblock.name.channel.set(name = '1', subBlock =
↳repcap.SubBlock.Default, channel = repcap.Channel.Default)
```

No command help available

param name

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.name.channel.clone()
```

6.17.16.1.12.9 RfbWidth

SCPI Commands

```
SENSe:POWer:ACHannel:SBlock<SubBlock>:RFBWidth
```

class RfbWidthCls

RfbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:RFBWidth
value: float = driver.sense.power.achannel.sblock.rfbWidth.get(subBlock =
↳repcap.SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

return

bandwidth: No help available

set(bandwidth: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:RFBWidth
driver.sense.power.achannel.sblock.rfbWidth.set(bandwidth = 1.0, subBlock =
↳repcap.SubBlock.Default)
```

No command help available

param bandwidth

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

6.17.16.1.12.10 Technology

class TechnologyCls

Technology commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.technology.clone()
```

Subgroups

6.17.16.1.12.11 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.sblock.technology.channel.repcap_channel_get()
driver.sense.power.achannel.sblock.technology.channel.repcap_channel_set(repcap.Channel.
↪ Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SBLOCK<SubBlock>:TECHnology:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(subBlock=SubBlock.Default, channel=Channel.Default) → TechnologyStandardA

```
# SCPI: [SENSe]:POWer:ACHannel:SBLOCK<sb>:TECHnology[:CHANnel<ch>]
value: enums.TechnologyStandardA = driver.sense.power.achannel.sblock.
↪ technology.channel.get(subBlock = repcap.SubBlock.Default, channel = repcap.
↪ Channel.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

standard: No help available

set(*standard: TechnologyStandardA, subBlock=SubBlock.Default, channel=Channel.Default*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:TECHnology[:CHANnel<ch>]
driver.sense.power.achannel.sblock.technology.channel.set(standard = enums.
↳ TechnologyStandardA.GSM, subBlock = repcap.SubBlock.Default, channel = repcap.
↳ Channel.Default)
```

No command help available

param standard

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.technology.channel.clone()
```

6.17.16.1.12.12 TxChannel**class TxChannelCls**

TxChannel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.sblock.txChannel.clone()
```

Subgroups**6.17.16.1.12.13 Count****SCPI Commands**

```
SENSe:POWer:ACHannel:SBlock<SubBlock>:TXChannel:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(subBlock=SubBlock.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:TXChannel:COUNT
value: float = driver.sense.power.achannel.sblock.txChannel.count.get(subBlock,
↳= repcap.SubBlock.Default)
```

No command help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

return

number: No help available

set(number: float, subBlock=SubBlock.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SBlock<sb>:TXChannel:COUNT
driver.sense.power.achannel.sblock.txChannel.count.set(number = 1.0, subBlock =
↳= repcap.SubBlock.Default)
```

No command help available

param number

No help available

param subBlock

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Sblock')

6.17.16.1.13 Spacing**class SpacingCls**

Spacing commands group definition. 8 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.clone()
```

Subgroups**6.17.16.1.13.1 Achannel****SCPI Commands**

```
SENSe:POWer:ACHannel:SPACing:ACHannel
```

class AchannelCls

Achannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing[:ACHannel]
value: float = driver.sense.power.achannel.spacing.achannel.get()
```

No command help available

```
return
    spacing: No help available
```

set(spacing: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing[:ACHannel]
driver.sense.power.achannel.spacing.achannel.set(spacing = 1.0)
```

No command help available

```
param spacing
    No help available
```

6.17.16.1.13.2 Alternate<Channel>**RepCap Settings**

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.spacing.alternate.repcap_channel_get()
driver.sense.power.achannel.spacing.alternate.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:ALternate<Channel>
```

class AlternateCls

Alternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:ALternate<ch>
value: float = driver.sense.power.achannel.spacing.alternate.get(channel = ↵
↵repcap.Channel.Default)
```

No command help available

```
param channel
    optional repeated capability selector. Default value: Ch1 (settable in the interface
    'Alternate')
```

```
return
    spacing: No help available
```

set(spacing: float, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:ALternate<ch>
driver.sense.power.achannel.spacing.alternate.set(spacing = 1.0, channel = ↵
↵repcap.Channel.Default)
```

No command help available

param spacing

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Alternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.alternate.clone()
```

6.17.16.1.13.3 Channel<Channel>

RepCap Settings

```
# Range: Ch1 .. Ch64
rc = driver.sense.power.achannel.spacing.channel.repcap_channel_get()
driver.sense.power.achannel.spacing.channel.repcap_channel_set(repcap.Channel.Ch1)
```

SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:CHANnel<Channel>
```

class ChannelCls

Channel commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Channel, default value after init: Channel.Ch1

get(channel=Channel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:CHANnel<ch>
value: float = driver.sense.power.achannel.spacing.channel.get(channel = repcap.
↵Channel.Default)
```

No command help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

return

spacing: No help available

set(spacing: float, channel=Channel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:CHANnel<ch>
driver.sense.power.achannel.spacing.channel.set(spacing = 1.0, channel = repcap.
↪Channel.Default)
```

No command help available

param spacing

No help available

param channel

optional repeated capability selector. Default value: Ch1 (settable in the interface 'Channel')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.channel.clone()
```

6.17.16.1.13.4 Gap<GapChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.sense.power.achannel.spacing.gap.repcap_gapChannel_get()
driver.sense.power.achannel.spacing.gap.repcap_gapChannel_set(repcap.GapChannel.Nr1)
```

class GapCls

Gap commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: GapChannel, default value after init: GapChannel.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.gap.clone()
```

Subgroups

6.17.16.1.13.5 Auto

SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:GAP<GapChannel>:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(spacing: float, gapChannel=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:GAP<gap>[:AUTO]
driver.sense.power.achannel.spacing.gap.auto.set(spacing = 1.0, gapChannel =
↳repcap.GapChannel.Default)
```

No command help available

param spacing

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.13.6 Manual

class ManualCls

Manual commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.gap.manual.clone()
```

Subgroups

6.17.16.1.13.7 Lower

SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:GAP<GapChannel>:MANual:LOWer
```

class LowerCls

Lower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(sb_gaps: SubBlockGaps, gapChannel=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:GAP<gap>:MANual:LOWer
value: float = driver.sense.power.achannel.spacing.gap.manual.lower.get(sb_gaps,
↳enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return

spacing: No help available

set(*sb_gaps*: SubBlockGaps, *spacing*: float, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:GAP<gap>:MANual:LOWer
driver.sense.power.achannel.spacing.gap.manual.lower.set(sb_gaps = enums.
↳ SubBlockGaps.AB, spacing = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param spacing
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.13.8 Upper

SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:GAP<GapChannel>:MANual:UPPer
```

class UpperCls

Upper commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sb_gaps*: SubBlockGaps, *gapChannel*=GapChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:GAP<gap>:MANual:UPPer
value: float = driver.sense.power.achannel.spacing.gap.manual.upper.get(sb_gaps,
↳ = enums.SubBlockGaps.AB, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param gapChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

return
spacing: No help available

set(*sb_gaps*: SubBlockGaps, *spacing*: float, *gapChannel*=GapChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:GAP<gap>:MANual:UPPer
driver.sense.power.achannel.spacing.gap.manual.upper.set(sb_gaps = enums.
↳ SubBlockGaps.AB, spacing = 1.0, gapChannel = repcap.GapChannel.Default)
```

No command help available

param sb_gaps
No help available

param spacing

No help available

param gapChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Gap')

6.17.16.1.13.9 UaChannel**SCPI Commands**

`SENSe:POWer:ACHannel:SPACing:UACHannel`

class UaChannelCls

UaChannel commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:UACHannel
value: float = driver.sense.power.achannel.spacing.uaChannel.get()
```

No command help available

return

spacing: No help available

set(spacing: float) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:UACHannel
driver.sense.power.achannel.spacing.uaChannel.set(spacing = 1.0)
```

No command help available

param spacing

No help available

6.17.16.1.13.10 Ualternate<UpperAltChannel>**RepCap Settings**

```
# Range: Nr1 .. Nr63
rc = driver.sense.power.achannel.spacing.ualternate.repcap_upperAltChannel_get()
driver.sense.power.achannel.spacing.ualternate.repcap_upperAltChannel_set(repcap.
↪UpperAltChannel.Nr1)
```


SCPI Commands

```
SENSe:POWer:ACHannel:SPACing:UALternate<UpperAltChannel>
```

class UalternateCls

Ualternate commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: UpperAltChannel, default value after init: UpperAltChannel.Nr1

get(upperAltChannel=UpperAltChannel.Default) → float

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:UALternate<ch>
value: float = driver.sense.power.achannel.spacing.ualternate.
↳get(upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

return

spacing: No help available

set(spacing: float, upperAltChannel=UpperAltChannel.Default) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SPACing:UALternate<ch>
driver.sense.power.achannel.spacing.ualternate.set(spacing = 1.0,
↳upperAltChannel = repcap.UpperAltChannel.Default)
```

No command help available

param spacing

No help available

param upperAltChannel

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ualternate')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.spacing.ualternate.clone()
```

6.17.16.1.14 Ssetup

SCPI Commands

```
SENSe:POWer:ACHannel:SSETup
```

class SsetupCls

Ssetup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:ACHannel:SSEtup
value: bool = driver.sense.power.achannel.ssetup.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:POWer:ACHannel:SSEtup
driver.sense.power.achannel.ssetup.set(state = False)
```

No command help available

param state
No help available

6.17.16.1.15 TxChannel

class TxChannelCls

TxChannel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.power.achannel.txChannel.clone()
```

Subgroups

6.17.16.1.15.1 Count

SCPI Commands

```
SENSe:POWer:ACHannel:TXChannel:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:ACHannel:TXChannel:COUNT
value: float = driver.sense.power.achannel.txChannel.count.get()
```

No command help available

return
number: No help available

set(*number: float*) → None

```
# SCPI: [SENSe]:POWer:ACHannel:TXChannel:COUNT
driver.sense.power.achannel.txChannel.count.set(number = 1.0)
```

No command help available

param number
No help available

6.17.16.2 Bandwidth

SCPI Commands

```
SENSe:POWer:BWIDth
```

class BandwidthCls

Bandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:BWIDth
value: float = driver.sense.power.bandwidth.get()
```

No command help available

return
percentage: No help available

set(*percentage: float*) → None

```
# SCPI: [SENSe]:POWer:BWIDth
driver.sense.power.bandwidth.set(percentage = 1.0)
```

No command help available

param percentage
No help available

6.17.16.3 Hspeed

SCPI Commands

```
SENSe:POWer:HSPeed
```

class HspeedCls

Hspeed commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:HSPeed
value: bool = driver.sense.power.hspeed.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: [SENSe]:POWer:HSPEED
driver.sense.power.hspeer.set(state = False)
```

No command help available

param state

No help available

6.17.16.4 Ncorrection

SCPI Commands

```
SENSe:POWer:NCORrection
```

class NcorrectionCls

Ncorrection commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:POWer:NCORrection
value: bool = driver.sense.power.ncorrection.get()
```

This command turns noise cancellation on and off. If noise cancellation is on, the R&S FSWP performs a reference measurement to determine its inherent noise and subtracts the result from the channel power measurement result (first active trace only) . For more information see ‘Noise Cancellation’.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: [SENSe]:POWer:NCORrection
driver.sense.power.ncorrection.set(state = False)
```

This command turns noise cancellation on and off. If noise cancellation is on, the R&S FSWP performs a reference measurement to determine its inherent noise and subtracts the result from the channel power measurement result (first active trace only) . For more information see ‘Noise Cancellation’.

param state

ON | OFF | 1 | 0

6.17.16.5 Trace

SCPI Commands

SENSe:POWer:TRACe

class TraceCls

Trace commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:POWer:TRACe
value: float = driver.sense.power.trace.get()
```

No command help available

return

trace_number: No help available

set(trace_number: float) → None

```
# SCPI: [SENSe]:POWer:TRACe
driver.sense.power.trace.set(trace_number = 1.0)
```

No command help available

param trace_number

No help available

6.17.17 Probe<Probe>

RepCap Settings

```
# Range: Nr1 .. Nr8
rc = driver.sense.probe.repcap_probe_get()
driver.sense.probe.repcap_probe_set(repcap.Probe.Nr1)
```

class ProbeCls

Probe commands group definition. 12 total commands, 2 Subgroups, 0 group commands Repeated Capability:

Probe, default value after init: Probe.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.probe.clone()
```

Subgroups

6.17.17.1 Id

class IdCls

Id commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.probe.id.clone()
```

Subgroups

6.17.17.1.1 PartNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:PARTnumber
```

class PartNumberCls

PartNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:ID:PARTnumber
value: float = driver.sense.probe.id.partNumber.get(probe = repcap.Probe.
↳Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

part_number: No help available

6.17.17.1.2 SrNumber

SCPI Commands

```
SENSe:PROBe<Probe>:ID:SRnumber
```

class SrNumberCls

SrNumber commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:ID:SRnumber
value: float = driver.sense.probe.id.srNumber.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

serial_no: No help available

6.17.17.2 Setup

class SetupCls

Setup commands group definition. 10 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.probe.setup.clone()
```

Subgroups

6.17.17.2.1 AttRatio

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:ATTRatio
```

class AttRatioCls

AttRatio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
value: float = driver.sense.probe.setup.attRatio.get(probe = repcap.Probe.
↳ Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

attenuation_ratio: No help available

set(attenuation_ratio: float, probe=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:ATTRatio
driver.sense.probe.setup.attRatio.set(attenuation_ratio = 1.0, probe = repcap.
↳ Probe.Default)
```

No command help available

param attenuation_ratio

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.2 CmOffset**SCPI Commands**

SENSe:PROBe<Probe>:SETup:CMOffset

class CmOffsetCls

CmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
value: float = driver.sense.probe.setup.cmOffset.get(probe = repcap.Probe.
↳Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

cm_offset: No help available

set(*cm_offset: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:CMOffset
driver.sense.probe.setup.cmOffset.set(cm_offset = 1.0, probe = repcap.Probe.
↳Default)
```

No command help available

param cm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.3 DmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:DMOOffset
```

class DmOffsetCls

DmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOOffset
value: float = driver.sense.probe.setup.dmOffset.get(probe = repcap.Probe.
↳Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

dm_offset: No help available

set(*dm_offset: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:DMOOffset
driver.sense.probe.setup.dmOffset.set(dm_offset = 1.0, probe = repcap.Probe.
↳Default)
```

No command help available

param dm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.4 Mode

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeSetupMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
value: enums.ProbeSetupMode = driver.sense.probe.setup.mode.get(probe = repcap.
↳Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(mode: ProbeSetupMode, probe=Probe.Default) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:MODE
driver.sense.probe.setup.mode.set(mode = enums.ProbeSetupMode.NOAction, probe = ↵
↵repcap.Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.5 Name

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(probe=Probe.Default) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NAME
value: float = driver.sense.probe.setup.name.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

name: No help available

6.17.17.2.6 NmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:NMOFfset
```

class NmOffsetCls

NmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset
value: float = driver.sense.probe.setup.nmOffset.get(probe = repcap.Probe.
↳Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

nm_offset: No help available

set(*nm_offset: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:NMOffset
driver.sense.probe.setup.nmOffset.set(nm_offset = 1.0, probe = repcap.Probe.
↳Default)
```

No command help available

param nm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.7 Pmode

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMODE
```

class PmodeCls

Pmode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → ProbeMode

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
value: enums.ProbeMode = driver.sense.probe.setup.pmode.get(probe = repcap.
↳Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

mode: No help available

set(*mode: ProbeMode, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMODE
driver.sense.probe.setup.pmode.set(mode = enums.ProbeMode.CM, probe = repcap.
↪Probe.Default)
```

No command help available

param mode

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.8 PmOffset

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:PMOffset
```

class PmOffsetCls

PmOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
value: float = driver.sense.probe.setup.pmOffset.get(probe = repcap.Probe.
↪Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

pm_offset: No help available

set(*pm_offset: float, probe=Probe.Default*) → None

```
# SCPI: [SENSe]:PROBe<pb>:SETup:PMOffset
driver.sense.probe.setup.pmOffset.set(pm_offset = 1.0, probe = repcap.Probe.
↪Default)
```

No command help available

param pm_offset

No help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

6.17.17.2.9 State

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:STATe
value: float = driver.sense.probe.setup.state.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

state: No help available

6.17.17.2.10 TypePy

SCPI Commands

```
SENSe:PROBe<Probe>:SETup:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*probe=Probe.Default*) → float

```
# SCPI: [SENSe]:PROBe<pb>:SETup:TYPE
value: float = driver.sense.probe.setup.typePy.get(probe = repcap.Probe.Default)
```

No command help available

param probe

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Probe')

return

type_py: No help available

6.17.18 Rlength

SCPI Commands

`SENSe:RLENgth`

class RlengthCls

Rlength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:RLENgth
value: float = driver.sense.rlength.get()
```

No command help available

```
return
    sample_count: No help available
```

6.17.19 Roscillator

class RoscillatorCls

Roscillator commands group definition. 12 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.roscillator.clone()
```

Subgroups

6.17.19.1 Coupling

class CouplingCls

Coupling commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.roscillator.coupling.clone()
```

Subgroups

6.17.19.1.1 Bandwidth

SCPI Commands

```
SENSe:ROSCillator:COUpling:BANDwidth
```

class BandwidthCls

Bandwidth commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ROSCillator:COUpling:BANDwidth
value: float = driver.sense.rosillator.coupling.bandwidth.get()
```

This command defines coupling bandwidth for the internal reference frequency.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.
- Select manual bandwidth mode ([SENSe:]ROSCillator:COUpling:BANDwidth:MODE)

return

bandwidth: numeric value Bandwidths 20 mHz, 1 Hz and 100 kHz are supported. Unit: Hz

set(bandwidth: float) → None

```
# SCPI: [SENSe]:ROSCillator:COUpling:BANDwidth
driver.sense.rosillator.coupling.bandwidth.set(bandwidth = 1.0)
```

This command defines coupling bandwidth for the internal reference frequency.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.
- Select manual bandwidth mode ([SENSe:]ROSCillator:COUpling:BANDwidth:MODE)

param bandwidth

numeric value Bandwidths 20 mHz, 1 Hz and 100 kHz are supported. Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.rosillator.coupling.bandwidth.clone()
```

Subgroups

6.17.19.1.1.1 Mode

SCPI Commands

SENSe:ROSCillator:COUPling:BANDwidth:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoManualMode

```
# SCPI: [SENSe]:ROSCillator:COUPling:BANDwidth:MODE
value: enums.AutoManualMode = driver.sense.rosillator.coupling.bandwidth.mode.
↳get()
```

This command selects coupling bandwidth mode for the internal reference frequency.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.

return

mode: AUTO Automatically selects an appropriate coupling bandwidth. MAN-ual Manual selection of coupling bandwidth. You can select the bandwidth with [SENSe:]ROSCillator:COUPling:BANDwidth.

set(mode: AutoManualMode) → None

```
# SCPI: [SENSe]:ROSCillator:COUPling:BANDwidth:MODE
driver.sense.rosillator.coupling.bandwidth.mode.set(mode = enums.
↳AutoManualMode.AUTO)
```

This command selects coupling bandwidth mode for the internal reference frequency.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.

param mode

AUTO Automatically selects an appropriate coupling bandwidth. MANual Manual selection of coupling bandwidth. You can select the bandwidth with [SENSe:]ROSCillator:COUPling:BANDwidth.

6.17.19.1.2 Mode

SCPI Commands

SENSe:ROSCillator:COUPling:MODE

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AutoMode

```
# SCPI: [SENSe]:ROSCillator:COUpling:MODE
value: enums.AutoMode = driver.sense.rosillator.coupling.mode.get()
```

This command turns the coupling of the internal reference frequency on and off.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.

return

state: AUTO Automatically turns the coupling on and off, depending on the current measurement scenario. OFF Decouples the reference frequencies. ON Couples the reference frequencies.

set(state: AutoMode) → None

```
# SCPI: [SENSe]:ROSCillator:COUpling:MODE
driver.sense.rosillator.coupling.mode.set(state = enums.AutoMode.AUTO)
```

This command turns the coupling of the internal reference frequency on and off.

INTRO_CMD_HELP: Prerequisites for this command

- Options R&S FSWP-B60 or -B61 must be available.

param state

AUTO Automatically turns the coupling on and off, depending on the current measurement scenario. OFF Decouples the reference frequencies. ON Couples the reference frequencies.

6.17.19.2 LbWidth

SCPI Commands

```
SENSe:ROSCillator:LBWidth
```

class LbWidthCls

LbWidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:ROSCillator:LBWidth
value: float = driver.sense.rosillator.lbWidth.get()
```

Defines the loop bandwidth, that is, the speed of internal synchronization with the reference frequency. The setting requires a compromise between performance and increasing phase noise. For a variable external reference frequency with a narrow tuning range (± 0.5 ppm), the loop bandwidth is fixed to 0.1 Hz and cannot be changed.

return

bandwidth: 0.1 Hz | 1 Hz | 3 Hz | 10 Hz | 30 Hz | 100 Hz | 300 Hz The possible values depend on the reference source and tuning range (see Table 'Available Reference Frequency Input'). Unit: Hz

set(*bandwidth: float*) → None

```
# SCPI: [SENSe]:ROSCillator:LBWidth
driver.sense.roscillator.lbWidth.set(bandwidth = 1.0)
```

Defines the loop bandwidth, that is, the speed of internal synchronization with the reference frequency. The setting requires a compromise between performance and increasing phase noise. For a variable external reference frequency with a narrow tuning range (+/- 0.5 ppm) , the loop bandwidth is fixed to 0.1 Hz and cannot be changed.

param bandwidth

0.1 Hz | 1 Hz | 3 Hz | 10 Hz | 30 Hz | 100 Hz | 300 Hz The possible values depend on the reference source and tuning range (see Table ‘Available Reference Frequency Input’).
Unit: Hz

6.17.19.3 O100

SCPI Commands

```
SENSe:ROSCillator:O100
```

class O100Cls

O100 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ROSCillator:O100
value: bool = driver.sense.roscillator.o100.get()
```

No command help available

return

state: No help available

set(*state: bool*) → None

```
# SCPI: [SENSe]:ROSCillator:O100
driver.sense.roscillator.o100.set(state = False)
```

No command help available

param state

No help available

6.17.19.4 O640

SCPI Commands

```
SENSe:ROSCillator:O640
```

class O640Cls

O640 commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ROSCillator:O640
value: bool = driver.sense.roscillator.o640.get()
```

This command turns the output of a reference signal on the corresponding connector ('Ref Output') on and off. [SENSe:]ROSCillator:O100: Provides a 100 MHz reference signal on corresponding connector. [SENSe:]ROSCillator:O640: Provides a 640 MHz reference signal on corresponding connector.

return

state: ON | OFF | 1 | 0 OFF | 0 Switches the reference off. ON | 1 Switches the reference on

set(state: bool) → None

```
# SCPI: [SENSe]:ROSCillator:O640
driver.sense.roscillator.o640.set(state = False)
```

This command turns the output of a reference signal on the corresponding connector ('Ref Output') on and off. [SENSe:]ROSCillator:O100: Provides a 100 MHz reference signal on corresponding connector. [SENSe:]ROSCillator:O640: Provides a 640 MHz reference signal on corresponding connector.

param state

ON | OFF | 1 | 0 OFF | 0 Switches the reference off. ON | 1 Switches the reference on

6.17.19.5 Osync

SCPI Commands

```
SENSe:ROSCillator:OSYNc
```

class OsyncCls

Osync commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:ROSCillator:OSYNc
value: bool = driver.sense.roscillator.osync.get()
```

If enabled, a 100 MHz reference signal is provided to the 'SYNC TRIGGER OUTPUT' connector.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: [SENSe]:ROSCillator:OSYNc
driver.sense.roscillator.osync.set(state = False)
```

If enabled, a 100 MHz reference signal is provided to the 'SYNC TRIGGER OUTPUT' connector.

param state

ON | OFF | 1 | 0

6.17.19.6 Output<OutputConnector>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.sense.roscillator.output.repcap_outputConnector_get()
driver.sense.roscillator.output.repcap_outputConnector_set(repcap.OutputConnector.Nr1)
```

SCPI Commands

```
SENSe:ROSCillator:OUTPut<OutputConnector>
```

class OutputCls

Output commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: OutputConnector, default value after init: OutputConnector.Nr1

get(outputConnector=OutputConnector.Default) → RoscillatorRefOut

```
# SCPI: [SENSe]:ROSCillator:OUTPut<o>
value: enums.RoscillatorRefOut = driver.sense.roscillator.output.
↳ get(outputConnector = repcap.OutputConnector.Default)
```

No command help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

return

ref_out: No help available

set(ref_out: RoscillatorRefOut, outputConnector=OutputConnector.Default) → None

```
# SCPI: [SENSe]:ROSCillator:OUTPut<o>
driver.sense.roscillator.output.set(ref_out = enums.RoscillatorRefOut.EXternal1,
↳ outputConnector = repcap.OutputConnector.Default)
```

No command help available

param ref_out

No help available

param outputConnector

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Output’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.rosillator.output.clone()
```

6.17.19.7 PassThrough

SCPI Commands

```
SENSe:ROSCillator:PASSthrough
```

class PassThroughCls

PassThrough commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: [SENSe]:ROSCillator:PASSthrough
value: str = driver.sense.rosillator.passThrough.get()
```

No command help available

```
return
    dev_name: No help available
```

set(dev_name: str, arg_1: bool) → None

```
# SCPI: [SENSe]:ROSCillator:PASSthrough
driver.sense.rosillator.passThrough.set(dev_name = '1', arg_1 = False)
```

No command help available

```
param dev_name
    No help available
```

```
param arg_1
    No help available
```

6.17.19.8 Source

SCPI Commands

```
SENSe:ROSCillator:SOURce
```

class SourceCls

Source commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → ReferenceSourceA

```
# SCPI: [SENSe]:ROSCillator:SOURce
value: enums.ReferenceSourceA = driver.sense.rosillator.source.get()
```

This command selects the reference oscillator. If you want to select the external reference, it must be connected to the R&S FSWP.

return

source: INTERNAL The internal reference is used (10 MHz) . EXTERNAL | EXTERNAL1 | EXT1 The external reference from the 'REF INPUT 10 MHZ' connector is used; if none is available, an error flag is displayed in the status bar. E10 The external reference from 'REF INPUT 1..50 MHZ' connector is used with a fixed 10 MHZ frequency; if none is available, an error flag is displayed in the status bar. E100 The external reference from the 'REF INPUT 100 MHZ / 1 GHz' connector is used with a fixed 100 MHZ frequency; if none is available, an error flag is displayed in the status bar. E1000 The external reference from 'REF INPUT 100 MHZ / 1 GHz' connector is used with a fixed 1 GHz frequency; if none is available, an error flag is displayed in the status bar. EAUTO The external reference is used as long as it is available, then the instrument switches to the internal reference. SYNC The external reference is used; if none is available, an error flag is displayed in the status bar.

set(source: ReferenceSourceA) → None

```
# SCPI: [SENSe]:ROSCillator:SOURce
driver.sense.roscillator.source.set(source = enums.ReferenceSourceA.E10)
```

This command selects the reference oscillator. If you want to select the external reference, it must be connected to the R&S FSWP.

param source

INTERNAL The internal reference is used (10 MHz) . EXTERNAL | EXTERNAL1 | EXT1 The external reference from the 'REF INPUT 10 MHZ' connector is used; if none is available, an error flag is displayed in the status bar. E10 The external reference from 'REF INPUT 1..50 MHZ' connector is used with a fixed 10 MHZ frequency; if none is available, an error flag is displayed in the status bar. E100 The external reference from the 'REF INPUT 100 MHZ / 1 GHz' connector is used with a fixed 100 MHZ frequency; if none is available, an error flag is displayed in the status bar. E1000 The external reference from 'REF INPUT 100 MHZ / 1 GHz' connector is used with a fixed 1 GHz frequency; if none is available, an error flag is displayed in the status bar. EAUTO The external reference is used as long as it is available, then the instrument switches to the internal reference. SYNC The external reference is used; if none is available, an error flag is displayed in the status bar.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.roscillator.source.clone()
```

Subgroups

6.17.19.8.1 Eauto

SCPI Commands

```
SENSe:ROSCillator:SOURce:EAUTO
```

class EautoCls

Eauto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → ReferenceSource

```
# SCPI: [SENSe]:ROSCillator:SOURce:EAUTO
value: enums.ReferenceSource = driver.sense.roscillator.source.eauto.get()
```

This command queries the current reference type in case you have activated an automatic switch to the internal reference if the external reference is missing.

return

reference: INT | EXT INT internal reference EXT external reference

6.17.19.9 Trange

SCPI Commands

```
SENSe:ROSCillator:TRANge
```

class TrangeCls

Trange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TuningRange

```
# SCPI: [SENSe]:ROSCillator:TRANge
value: enums.TuningRange = driver.sense.roscillator.trange.get()
```

Defines the tuning range. The tuning range is only available for the variable external reference frequency. It determines how far the frequency may deviate from the defined level in parts per million (10-6) . For more information see Table 'Available Reference Frequency Input'.

return

range_py: WIDE | SMALL The possible values depend on the reference source (see Table 'Available Reference Frequency Input') . SMALL With this smaller deviation (+/- 0.5 ppm) a very narrow fixed loop bandwidth of 0.1 Hz is realized. With this setting the instrument can synchronize to an external reference signal with a very precise frequency. Due to the very narrow loop bandwidth, unwanted noise or spurious components on the external reference input signal are strongly attenuated. Furthermore, the loop requires about 30 seconds to reach a locked state. During this locking process, 'NO REF' is displayed in the status bar. WIDE The larger deviation (+/- 6 ppm) allows the instrument to synchronize to less precise external reference input signals.

set(range_py: TuningRange) → None

```
# SCPI: [SENSe]:ROSCillator:TRANge
driver.sense.roscillator.trange.set(range_py = enums.TuningRange.SMALL)
```

Defines the tuning range. The tuning range is only available for the variable external reference frequency. It determines how far the frequency may deviate from the defined level in parts per million (10-6) . For more information see Table 'Available Reference Frequency Input'.

param range_py

WIDE | SMALL The possible values depend on the reference source (see Table 'Available Reference Frequency Input') . SMALL With this smaller deviation (+/- 0.5 ppm) a very narrow fixed loop bandwidth of 0.1 Hz is realized. With this setting the instrument can synchronize to an external reference signal with a very precise frequency. Due to the very narrow loop bandwidth, unwanted noise or spurious components on

the external reference input signal are strongly attenuated. Furthermore, the loop requires about 30 seconds to reach a locked state. During this locking process, 'NO REF' is displayed in the status bar. WIDE The larger deviation (+/- 6 ppm) allows the instrument to synchronize to less precise external reference input signals.

6.17.20 Rtms

class RtmsCls

Rtms commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.rtms.clone()
```

Subgroups

6.17.20.1 Capture

class CaptureCls

Capture commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.rtms.capture.clone()
```

Subgroups

6.17.20.1.1 Offset

SCPI Commands

```
SENSe:RTMS:CAPTure:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:RTMS:CAPTure:OFFSet
value: float = driver.sense.rtms.capture.offset.get()
```

No command help available

return
time: No help available

set(time: float) → None

```
# SCPI: [SENSe]:RTMS:CAPTure:OFFSet
driver.sense.rtms.capture.offset.set(time = 1.0)
```

No command help available

param time

No help available

6.17.21 Sampling

class SamplingCls

Sampling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sampling.clone()
```

Subgroups

6.17.21.1 Clkio

class ClkioCls

Clkio commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sampling.clkio.clone()
```

Subgroups

6.17.21.1.1 Output

SCPI Commands

```
SENSe:SAMpling:CLKio:OUTPut
```

class OutputCls

Output commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SAMpling:CLKio:OUTPut
value: bool = driver.sense.sampling.clkio.output.get()
```

No command help available

return

arg_1: No help available

set(*dev_name: str, arg_1: bool*) → None

```
# SCPI: [SENSe]:SAMPling:CLKio:OUTPut
driver.sense.sampling.clkio.output.set(dev_name = '1', arg_1 = False)
```

No command help available

param dev_name

No help available

param arg_1

No help available

6.17.22 SwapIq

SCPI Commands

```
SENSe:SWAPiq
```

class SwapIqCls

SwapIq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWAPiq
value: bool = driver.sense.swapIq.get()
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

return

arg_0: No help available

set(*arg_0: bool*) → None

```
# SCPI: [SENSe]:SWAPiq
driver.sense.swapIq.set(arg_0 = False)
```

This command defines whether or not the recorded I/Q pairs should be swapped (I<->Q) before being processed. Swapping I and Q inverts the sideband. This is useful if the DUT interchanged the I and Q parts of the signal; then the R&S FSWP can do the same to compensate for it. For GSM measurements: Try this function if the TSC can not be found.

param arg_0

ON | 1 I and Q signals are interchanged Inverted sideband, Q+j*I OFF | 0 I and Q signals are not interchanged Normal sideband, I+j*Q

6.17.23 Sweep

class SweepCls

Sweep commands group definition. 31 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.clone()
```

Subgroups

6.17.23.1 Count

SCPI Commands

```
SENSe:SWEEp:COUNT
```

class CountCls

Count commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEp:COUNT
value: float = driver.sense.sweep.count.get()
```

This command defines the number of measurements that the application uses to average traces. In continuous measurement mode, the application calculates the moving average over the average count. In single measurement mode, the application stops the measurement and calculates the average after the average count has been reached.

return

sweep_count: When you set a sweep count of 0 or 1, the R&S FSWP performs one single measurement in single measurement mode. In continuous measurement mode, if the sweep count is set to 0, a moving average over 10 measurements is performed. Range: 0 to 200000

set(sweep_count: float) → None

```
# SCPI: [SENSe]:SWEEp:COUNT
driver.sense.sweep.count.set(sweep_count = 1.0)
```

This command defines the number of measurements that the application uses to average traces. In continuous measurement mode, the application calculates the moving average over the average count. In single measurement mode, the application stops the measurement and calculates the average after the average count has been reached.

param sweep_count

When you set a sweep count of 0 or 1, the R&S FSWP performs one single measurement in single measurement mode. In continuous measurement mode, if the sweep count is set to 0, a moving average over 10 measurements is performed. Range: 0 to 200000

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.count.clone()
```

Subgroups

6.17.23.1.1 Current

SCPI Commands

```
SENSe:SWEEp:COUNT:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEp:COUNT:CURRent
value: float = driver.sense.sweep.count.current.get()
```

This query returns the current number of started sweeps or measurements. This command is only available if a sweep count value is defined and the instrument is in single sweep mode.

return
current_count: No help available

6.17.23.2 Duration

SCPI Commands

```
SENSe:SWEEp:DURation
```

class DurationCls

Duration commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEp:DURation
value: float = driver.sense.sweep.duration.get()
```

No command help available

return
time: No help available

6.17.23.3 Egate

SCPI Commands

```
SENSe:SWEEp:EGATe
```

class EgateCls

Egate commands group definition. 15 total commands, 8 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEEp:EGATe
value: bool = driver.sense.sweep.egate.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: [SENSe]:SWEEp:EGATe
driver.sense.sweep.egate.set(state = False)
```

No command help available

```
param state
    No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.clone()
```

Subgroups

6.17.23.3.1 Auto

SCPI Commands

```
SENSe:SWEEp:EGATe:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:AUTO
driver.sense.sweep.egate.auto.set(state = False)
```

No command help available

```
param state
    No help available
```

6.17.23.3.2 Holdoff

SCPI Commands

`SENSe:SWEep:EGATe:HOLDoff`

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:HOLDoff
value: float = driver.sense.sweep.egate.holdoff.get()
```

This command defines the gate delay time.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.
- Turn off automatic pulse detection ([SENSe:]SWEep:PULSe:DETection) .

return

delay_time: No help available

set(delay_time: float) → None

```
# SCPI: [SENSe]:SWEep:EGATe:HOLDoff
driver.sense.sweep.egate.holdoff.set(delay_time = 1.0)
```

This command defines the gate delay time.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.
- Turn off automatic pulse detection ([SENSe:]SWEep:PULSe:DETection) .

param delay_time

numeric value Unit: s

6.17.23.3.3 Length

SCPI Commands

`SENSe:SWEep:EGATe:LENGth`

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:LENGth
value: float = driver.sense.sweep.egate.length.get()
```

This command defines the gate length.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.
- Turn off automatic pulse detection ([SENSe:]SWEep:PULSe:DETection) .
- Select gate mode ‘Edge’ ([SENSe:]SWEep:EGATe:TYPE) .

return

gate_length: No help available

set(gate_length: float) → None

```
# SCPI: [SENSe]:SWEep:EGATe:LENGth
driver.sense.sweep.egate.length.set(gate_length = 1.0)
```

This command defines the gate length.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.
- Turn off automatic pulse detection ([SENSe:]SWEep:PULSe:DETection) .
- Select gate mode ‘Edge’ ([SENSe:]SWEep:EGATe:TYPE) .

param gate_length

numeric value Unit: s

6.17.23.3.4 Level

class LevelCls

Level commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.level.clone()
```

Subgroups

6.17.23.3.4.1 External<ExternalPort>

RepCap Settings

```
# Range: Nr1 .. Nr3
rc = driver.sense.sweep.egate.level.external.repcap_externalPort_get()
driver.sense.sweep.egate.level.external.repcap_externalPort_set(repcap.ExternalPort.Nr1)
```

SCPI Commands

```
SENSe:SWEEp:EGATe:LEVel:EXTeRnal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(externalPort=ExternalPort.Default) → float

```
# SCPI: [SENSe]:SWEEp:EGATe:LEVel[:EXTeRnal<tp>]
value: float = driver.sense.sweep.egate.level.external.get(externalPort = ↵
↵repcap.ExternalPort.Default)
```

No command help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

gate_level: No help available

set(gate_level: float, externalPort=ExternalPort.Default) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:LEVel[:EXTeRnal<tp>]
driver.sense.sweep.egate.level.external.set(gate_level = 1.0, externalPort = ↵
↵repcap.ExternalPort.Default)
```

No command help available

param gate_level

No help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.level.external.clone()
```

6.17.23.3.4.2 IfPower

SCPI Commands

```
SENSe:SWEEp:EGATe:LEVel:IFPower
```

class IfPowerCls

IfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:LEVel:IFPower
value: float = driver.sense.sweep.egate.level.ifPower.get()
```

No command help available

```
return
    gate_level: No help available
```

set(gate_level: float) → None

```
# SCPI: [SENSe]:SWEep:EGATe:LEVel:IFPower
driver.sense.sweep.egate.level.ifPower.set(gate_level = 1.0)
```

No command help available

```
param gate_level
    No help available
```

6.17.23.3.4.3 RfPower

SCPI Commands

```
SENSe:SWEep:EGATe:LEVel:RFPower
```

class RfPowerCls

RfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:EGATe:LEVel:RFPower
value: float = driver.sense.sweep.egate.level.rfPower.get()
```

No command help available

```
return
    gate_level: No help available
```

set(gate_level: float) → None

```
# SCPI: [SENSe]:SWEep:EGATe:LEVel:RFPower
driver.sense.sweep.egate.level.rfPower.set(gate_level = 1.0)
```

No command help available

```
param gate_level
    No help available
```

6.17.23.3.5 Polarity

SCPI Commands

SENSe:SWEEp:EGATe:POLarity

class PolarityCls

Polarity commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SlopeType

```
# SCPI: [SENSe]:SWEEp:EGATe:POLarity
value: enums.SlopeType = driver.sense.sweep.egate.polarity.get()
```

No command help available

```
    return
        polarity: No help available
```

set(polarity: SlopeType) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:POLarity
driver.sense.sweep.egate.polarity.set(polarity = enums.SlopeType.NEGative)
```

No command help available

```
    param polarity
        No help available
```

6.17.23.3.6 Source

SCPI Commands

SENSe:SWEEp:EGATe:SOURce

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerSourceD

```
# SCPI: [SENSe]:SWEEp:EGATe:SOURce
value: enums.TriggerSourceD = driver.sense.sweep.egate.source.get()
```

No command help available

```
    return
        source: No help available
```

set(source: TriggerSourceD) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:SOURce
driver.sense.sweep.egate.source.set(source = enums.TriggerSourceD.EXT2)
```

No command help available

param source
No help available

6.17.23.3.7 Trace<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.sense.sweep.egate.trace.repcap_trace_get()
driver.sense.sweep.egate.trace.repcap_trace_set(repcap.Trace.Tr1)
```

class TraceCls

Trace commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.trace.clone()
```

Subgroups

6.17.23.3.7.1 Comment

SCPI Commands

```
SENSe:SWEep:EGATe:TRACe<Trace>:COMMeNt
```

class CommentCls

Comment commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace=Trace.Default) → str

```
# SCPI: [SENSe]:SWEep:EGATe:TRACe<t>:COMMeNt
value: str = driver.sense.sweep.egate.trace.comment.get(trace = repcap.Trace.
↳Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

comment: No help available

set(comment: str, trace=Trace.Default) → None

```
# SCPI: [SENSe]:SWEep:EGATe:TRACe<t>:COMMeNt
driver.sense.sweep.egate.trace.comment.set(comment = '1', trace = repcap.Trace.
↳Default)
```

No command help available

param comment

No help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.17.23.3.7.2 Period

SCPI Commands

`SENSe:SWEep:EGATe:TRACe<Trace>:PERiod`

class PeriodCls

Period commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace=Trace.Default*) → float

```
# SCPI: [SENSe]:SWEep:EGATe:TRACe<t>:PERiod
value: float = driver.sense.sweep.egate.trace.period.get(trace = repcap.Trace.
↳Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

return

length: No help available

set(*length: float, trace=Trace.Default*) → None

```
# SCPI: [SENSe]:SWEep:EGATe:TRACe<t>:PERiod
driver.sense.sweep.egate.trace.period.set(length = 1.0, trace = repcap.Trace.
↳Default)
```

No command help available

param length

No help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

6.17.23.3.7.3 Start

SCPI Commands

```
SENSe:SWEEp:EGATe:TRACe<Trace>:STARt<GateRange>
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace*=Trace.Default, *gateRange*=GateRange.Nr1) → float

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>:STARt<gr>
value: float = driver.sense.sweep.egate.trace.start.get(trace = repcap.Trace.
↳Default, gateRange = repcap.GateRange.Nr1)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param gateRange

optional repeated capability selector. Default value: Nr1

return

time: No help available

set(*time*: float, *trace*=Trace.Default, *gateRange*=GateRange.Nr1) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>:STARt<gr>
driver.sense.sweep.egate.trace.start.set(time = 1.0, trace = repcap.Trace.
↳Default, gateRange = repcap.GateRange.Nr1)
```

No command help available

param time

No help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param gateRange

optional repeated capability selector. Default value: Nr1

6.17.23.3.7.4 State<Status>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.sense.sweep.egate.trace.state.repcap_status_get()
driver.sense.sweep.egate.trace.state.repcap_status_set(repcap.Status.Nr1)
```

SCPI Commands

`SENSe:SWEEp:EGATe:TRACe<Trace>:STATe<Status>`

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Status, default value after init: Status.Nr1

get(*trace*=Trace.Default, *status*=Status.Default) → bool

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>[:STATe<gr>]
value: bool = driver.sense.sweep.egate.trace.state.get(trace = repcap.Trace.
↳Default, status = repcap.Status.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param status

optional repeated capability selector. Default value: Nr1 (settable in the interface 'State')

return

state: No help available

set(*state*: bool, *trace*=Trace.Default, *status*=Status.Default) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>[:STATe<gr>]
driver.sense.sweep.egate.trace.state.set(state = False, trace = repcap.Trace.
↳Default, status = repcap.Status.Default)
```

No command help available

param state

No help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param status

optional repeated capability selector. Default value: Nr1 (settable in the interface 'State')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.trace.state.clone()
```

6.17.23.3.7.5 Stop<GateRange>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.sense.sweep.egate.trace.stop.repcap_gateRange_get()
driver.sense.sweep.egate.trace.stop.repcap_gateRange_set(repcap.GateRange.Nr1)
```

SCPI Commands

```
SENSe:SWEEp:EGATe:TRACe<Trace>:STOP<GateRange>
```

class StopCls

Stop commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: GateRange, default value after init: GateRange.Nr1

get(trace=Trace.Default, gateRange=GateRange.Default) → float

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>:STOP<gr>
value: float = driver.sense.sweep.egate.trace.stop.get(trace = repcap.Trace.
↳Default, gateRange = repcap.GateRange.Default)
```

No command help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param gateRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stop')

return

time: No help available

set(time: float, trace=Trace.Default, gateRange=GateRange.Default) → None

```
# SCPI: [SENSe]:SWEEp:EGATe:TRACe<t>:STOP<gr>
driver.sense.sweep.egate.trace.stop.set(time = 1.0, trace = repcap.Trace.
↳Default, gateRange = repcap.GateRange.Default)
```

No command help available

param time

No help available

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Trace')

param gateRange

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Stop')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.egate.trace.stop.clone()
```

6.17.23.3.8 TypePy

SCPI Commands

```
SENSe:SWEEP:EGATE:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → EgateType

```
# SCPI: [SENSe]:SWEEP:EGATE:TYPE
value: enums.EgateType = driver.sense.sweep.egate.typePy.get()
```

This command selects the gate type.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.

return

type_py: No help available

set(type_py: EgateType) → None

```
# SCPI: [SENSe]:SWEEP:EGATE:TYPE
driver.sense.sweep.egate.typePy.set(type_py = enums.EgateType.EDGE)
```

This command selects the gate type.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurement application.

param type_py

EDGE The gate opens when the gate level has been exceeded and closes when the time defined by the gate length has elapsed. LEVel The gate opens when the gate level has been exceeded and closes when the signal level again falls below the gate level. OFF The gate is off.

6.17.23.4 Mode

SCPI Commands

```
SENSe:SWEep:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SweepModeC

```
# SCPI: [SENSe]:SWEep:MODE
value: enums.SweepModeC = driver.sense.sweep.mode.get()
```

This command selects the configuration mode of the half decade table.

return

mode: MANual Manual mode: allows you to select a custom resolution bandwidth and number of cross-correlations for each half decade. • Define the RBW for a half decade with [SENSe:]LIST:RANGeri:BWIDth[:RESolution]. • Define the number of cross-correlations for a half decade with [SENSe:]LIST:RANGeri:XCOunt. NORMal Automatic mode: the application selects the resolution bandwidth and number of cross-correlations based on the RBW and XCORR factors. • Define the RBW factor with [SENSe:]LIST:BWIDth[:RESolution]:RATio. • Define the XCORR factor with [SENSe:]SWEep:XFACTOR. FAST Sets mode to NORMal and XCORR Count to 1. Only available remote. AVERaged Sets mode to NORMal and XCORR Count to 10. Only available remote.

set(mode: SweepModeC) → None

```
# SCPI: [SENSe]:SWEep:MODE
driver.sense.sweep.mode.set(mode = enums.SweepModeC.AUTO)
```

This command selects the configuration mode of the half decade table.

param mode

MANual Manual mode: allows you to select a custom resolution bandwidth and number of cross-correlations for each half decade. • Define the RBW for a half decade with [SENSe:]LIST:RANGeri:BWIDth[:RESolution]. • Define the number of cross-correlations for a half decade with [SENSe:]LIST:RANGeri:XCOunt. NORMal Automatic mode: the application selects the resolution bandwidth and number of cross-correlations based on the RBW and XCORR factors. • Define the RBW factor with [SENSe:]LIST:BWIDth[:RESolution]:RATio. • Define the XCORR factor with [SENSe:]SWEep:XFACTOR. FAST Sets mode to NORMal and XCORR Count to 1. Only available remote. AVERaged Sets mode to NORMal and XCORR Count to 10. Only available remote.

6.17.23.5 Optimize

SCPI Commands

SENSe:SWEEP:OPTimize

class OptimizeCls

Optimize commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SweepOptimize

```
# SCPI: [SENSe]:SWEEP:OPTimize
value: enums.SweepOptimize = driver.sense.sweep.optimize.get()
```

No command help available

return
mode: No help available

set(mode: SweepOptimize) → None

```
# SCPI: [SENSe]:SWEEP:OPTimize
driver.sense.sweep.optimize.set(mode = enums.SweepOptimize.AUTO)
```

No command help available

param mode
No help available

6.17.23.6 Points

SCPI Commands

SENSe:SWEEP:POINTs

class PointsCls

Points commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:SWEEP:POINTs
value: int = driver.sense.sweep.points.get()
```

No command help available

return
sweep_points: No help available

set(sweep_points: int) → None

```
# SCPI: [SENSe]:SWEEP:POINTs
driver.sense.sweep.points.set(sweep_points = 1)
```

No command help available

param sweep_points
No help available

6.17.23.7 Scapture

class ScaptureCls

Scapture commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.scapture.clone()
```

Subgroups

6.17.23.7.1 Events

SCPI Commands

```
SENSe:SWEEp:SCAPture:EVENTs
```

class EventsCls

Events commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:SWEEp:SCAPture:EVENTs
value: int = driver.sense.sweep.scapture.events.get()
```

No command help available

```
return
    count: No help available
```

set(count: int) → None

```
# SCPI: [SENSe]:SWEEp:SCAPture:EVENTs
driver.sense.sweep.scapture.events.set(count = 1)
```

No command help available

```
param count
    No help available
```

6.17.23.7.2 Gap

SCPI Commands

```
SENSe:SWEEp:SCAPture:GAP
```

class GapCls

Gap commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: [SENSe]:SWEep:SCAPture:GAP
value: int = driver.sense.sweep.scapture.gap.get()
```

No command help available

```
return
seg_cap_gap_len: No help available
```

set(seg_cap_gap_len: int) → None

```
# SCPI: [SENSe]:SWEep:SCAPture:GAP
driver.sense.sweep.scapture.gap.set(seg_cap_gap_len = 1)
```

No command help available

```
param seg_cap_gap_len
No help available
```

6.17.23.7.3 Length

class LengthCls

Length commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.scapture.length.clone()
```

Subgroups

6.17.23.7.3.1 Time

SCPI Commands

```
SENSe:SWEep:SCAPture:LENGth:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEep:SCAPture:LENGth[:TIME]
value: float = driver.sense.sweep.scapture.length.time.get()
```

No command help available

```
return
segment_len: No help available
```

set(*segment_len: float*) → None

```
# SCPI: [SENSe]:SWEEP:SCAPture:LENGth[:TIME]
driver.sense.sweep.scapture.length.time.set(segment_len = 1.0)
```

No command help available

param segment_len

No help available

6.17.23.7.4 Offset

class OffsetCls

Offset commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.scapture.offset.clone()
```

Subgroups

6.17.23.7.4.1 Time

SCPI Commands

```
SENSe:SWEEP:SCAPture:OFFSet:TIME
```

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEP:SCAPture:OFFSet[:TIME]
value: float = driver.sense.sweep.scapture.offset.time.get()
```

No command help available

return

pretrigger: No help available

set(*pretrigger: float*) → None

```
# SCPI: [SENSe]:SWEEP:SCAPture:OFFSet[:TIME]
driver.sense.sweep.scapture.offset.time.set(pretrigger = 1.0)
```

No command help available

param pretrigger

No help available

6.17.23.7.5 State

SCPI Commands

SENSe:SWEEP:SCAPture:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: [SENSe]:SWEEP:SCAPture[:STATe]
value: bool = driver.sense.sweep.scapture.state.get()
```

No command help available

```
return
    seg_cap_state: No help available
```

set(seg_cap_state: bool) → None

```
# SCPI: [SENSe]:SWEEP:SCAPture[:STATe]
driver.sense.sweep.scapture.state.set(seg_cap_state = False)
```

No command help available

```
param seg_cap_state
    No help available
```

6.17.23.8 Time

SCPI Commands

SENSe:SWEEP:TIME

class TimeCls

Time commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEP:TIME
value: float = driver.sense.sweep.time.get()
```

This command defines the measurement time. It automatically decouples the time from any other settings.

```
return
    time: refer to data sheet Unit: S
```

set(time: float) → None

```
# SCPI: [SENSe]:SWEEP:TIME
driver.sense.sweep.time.set(time = 1.0)
```

This command defines the measurement time. It automatically decouples the time from any other settings.

param time
refer to data sheet Unit: S

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.time.clone()
```

Subgroups

6.17.23.8.1 Auto

SCPI Commands

```
SENSe:SWEEP:TIME:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool) → None

```
# SCPI: [SENSe]:SWEEP:TIME:AUTO
driver.sense.sweep.time.auto.set(state = False)
```

No command help available

param state
No help available

6.17.23.9 TypePy

SCPI Commands

```
SENSe:SWEEP:TYPE
```

class TypePyCls

TypePy commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → SweepType

```
# SCPI: [SENSe]:SWEEP:TYPE
value: enums.SweepType = driver.sense.sweep.typePy.get()
```

No command help available

return
type_py: No help available

set(type_py: SweepType) → None

```
# SCPI: [SENSe]:SWEEP:TYPE
driver.sense.sweep.typePy.set(type_py = enums.SweepType.AUTO)
```

No command help available

param type_py
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.typePy.clone()
```

Subgroups

6.17.23.9.1 Used

SCPI Commands

```
SENSe:SWEEP:TYPE:USED
```

class UsedCls

Used commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SWEEP:TYPE:USED
value: float = driver.sense.sweep.typePy.used.get()
```

No command help available

return
type_py: No help available

6.17.23.10 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.sense.sweep.window.repcap_window_get()
driver.sense.sweep.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.sweep.window.clone()
```

Subgroups

6.17.23.10.1 Points

SCPI Commands

```
SENSe:SWEEp:WINDow<Window>:POINts
```

class PointsCls

Points commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → float

```
# SCPI: [SENSe]:SWEEp[:WINDow<n>]:POINts
value: float = driver.sense.sweep.window.points.get(window = repcap.Window.
↳Default)
```

This command defines the number of measurement points to analyze after a measurement.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

return

no_points: No help available

set(no_points: float, window=Window.Default) → None

```
# SCPI: [SENSe]:SWEEp[:WINDow<n>]:POINts
driver.sense.sweep.window.points.set(no_points = 1.0, window = repcap.Window.
↳Default)
```

This command defines the number of measurement points to analyze after a measurement.

param no_points

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

6.17.24 SymbolRate

SCPI Commands

`SENSe:SRATe`

class SymbolRateCls

SymbolRate commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: [SENSe]:SRATe
value: float = driver.sense.symbolRate.get()
```

No command help available

```
return
    sample_rate: No help available
```

6.17.25 Trace

class TraceCls

Trace commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.trace.clone()
```

Subgroups

6.17.25.1 Iq

class IqCls

Iq commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.trace.iq.clone()
```

Subgroups

6.17.25.1.1 Sync

class SyncCls

Sync commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.trace.iq.sync.clone()
```

Subgroups

6.17.25.1.1.1 Mode

SCPI Commands

```
SENSe:TRACe:IQ:SYNC:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → MsSyncMode

```
# SCPI: [SENSe]:TRACe:IQ:SYNC:MODE
value: enums.MsSyncMode = driver.sense.trace.iq.sync.mode.get()
```

No command help available

```
return
    ms_sync_mode: No help available
```

set(dev_name: str, ms_sync_mode: MsSyncMode) → None

```
# SCPI: [SENSe]:TRACe:IQ:SYNC:MODE
driver.sense.trace.iq.sync.mode.set(dev_name = '1', ms_sync_mode = enums.
    ↪ MsSyncMode.MASTer)
```

No command help available

```
param dev_name
    No help available
```

```
param ms_sync_mode
    No help available
```

6.17.26 Window<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.sense.window.repcap_window_get()
driver.sense.window.repcap_window_set(repcap.Window.Nr1)
```

class WindowCls

Window commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.window.clone()
```

Subgroups

6.17.26.1 Detector<Trace>

RepCap Settings

```
# Range: Tr1 .. Tr16
rc = driver.sense.window.detector.repcap_trace_get()
driver.sense.window.detector.repcap_trace_set(repcap.Trace.Tr1)
```

class DetectorCls

Detector commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Tr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.window.detector.clone()
```

Subgroups

6.17.26.1.1 Function

SCPI Commands

```
SENSe:WINDow<Window>:DETEctor<Trace>:FUNctIon
```

class FunctionCls

Function commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(window=Window.Default, trace=Trace.Default) → DetectorB

```
# SCPI: [SENSe][:WINDow<n>]:DETEctor<t>[:FUNction]
value: enums.DetectorB = driver.sense.window.detector.function.get(window = ↵
↵repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

return

detector: No help available

set(detector: DetectorB, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: [SENSe][:WINDow<n>]:DETEctor<t>[:FUNction]
driver.sense.window.detector.function.set(detector = enums.DetectorB.ACSine, ↵
↵window = repcap.Window.Default, trace = repcap.Trace.Default)
```

No command help available

param detector

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.window.detector.function.clone()
```

Subgroups

6.17.26.1.1.1 Auto

SCPI Commands

```
SENSe:WINDow<Window>:DETEctor<Trace>:FUNction:AUTO
```

class AutoCls

Auto commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(state: bool, window=Window.Default, trace=Trace.Default) → None

```
# SCPI: [SENSe][:WINDow<n>]:DETeCTOR<t>[:FUNction]:AUTO
driver.sense.window.detector.function.auto.set(state = False, window = repcap.
↪Window.Default, trace = repcap.Trace.Default)
```

No command help available

param state

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Window')

param trace

optional repeated capability selector. Default value: Tr1 (settable in the interface 'Detector')

6.18 Source

class SourceCls

Source commands group definition. 45 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.clone()
```

Subgroups

6.18.1 Current

class CurrentCls

Current commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.clone()
```

Subgroups

6.18.1.1 Auxiliary

class AuxiliaryCls

Auxiliary commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.auxiliary.clone()
```

Subgroups

6.18.1.1.1 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.auxiliary.limit.clone()
```

Subgroups

6.18.1.1.1.1 High

SCPI Commands

```
SOURce:CURRent:AUX:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:CURRent:AUX:LIMit:HIGH
value: float = driver.source.current.auxiliary.limit.high.get()
```

This command returns the maximum current of the Vaux connector.

return

current: numeric value The return value is always 0.1 A.

set(current: float) → None

```
# SCPI: SOURce:CURRent:AUX:LIMit:HIGH
driver.source.current.auxiliary.limit.high.set(current = 1.0)
```

This command returns the maximum current of the Vaux connector.

param current

numeric value The return value is always 0.1 A.

6.18.1.2 Control<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.source.current.control.repcap_source_get()
driver.source.current.control.repcap_source_set(repcap.Source.Nr1)
```

class ControlCls

Control commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.control.clone()
```

Subgroups

6.18.1.2.1 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.control.limit.clone()
```

Subgroups

6.18.1.2.1.1 High

SCPI Commands

```
SOURce:CURRent:CONTRol<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURCE:CURRENT:CONTROL<1/2>:LIMIT:HIGH
value: float = driver.source.current.control.limit.high.get(source = repcap.
↳ Source.Default)
```

This command returns the maximum current of the Vtune connector.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

current: numeric value The return value is always 0.02 A. Unit: A

set(current: float, source=Source.Default) → None

```
# SCPI: SOURCE:CURRENT:CONTROL<1/2>:LIMIT:HIGH
driver.source.current.control.limit.high.set(current = 1.0, source = repcap.
↳ Source.Default)
```

This command returns the maximum current of the Vtune connector.

param current

numeric value The return value is always 0.02 A. Unit: A

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.18.1.3 Power<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.source.current.power.repcap_source_get()
driver.source.current.power.repcap_source_set(repcap.Source.Nr1)
```

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.power.clone()
```

Subgroups

6.18.1.3.1 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.power.limit.clone()
```

Subgroups

6.18.1.3.1.1 High

SCPI Commands

```
SOURce:CURRent:POWer<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:CURRent:POWer<1|2>:LIMit:HIGH
value: float = driver.source.current.power.limit.high.get(source = repcap.
↳ Source.Default)
```

This command defines the maximum current of the Vsupply connector.

INTRO_CMD_HELP: Prerequisites for this command

- Vsupply is controlled in terms of voltage (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Power.Level.Mode.set) .

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Power’)

return

current: numeric value Range: 0 to 2, Unit: A

set(current: float, source=Source.Default) → None

```
# SCPI: SOURce:CURRent:POWer<1|2>:LIMit:HIGH
driver.source.current.power.limit.high.set(current = 1.0, source = repcap.
↳ Source.Default)
```

This command defines the maximum current of the Vsupply connector.

INTRO_CMD_HELP: Prerequisites for this command

- Vsupply is controlled in terms of voltage (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Power.Level.Mode.set) .

param current

numeric value Range: 0 to 2, Unit: A

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.1.4 Sequence**class SequenceCls**

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.current.sequence.clone()
```

Subgroups**6.18.1.4.1 Result****SCPI Commands**

```
SOURce:CURRent:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Current_Vsupply: float: No parameter help available
- Current_Vtune: float: No parameter help available
- Current_Vaux: float: No parameter help available

get() → GetStruct

```
# SCPI: SOURce:CURRent:SEquence:RESult
value: GetStruct = driver.source.current.sequence.result.get()
```

This command queries the actually measured current on the DC power sources.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on the DC power source (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)
-

return

structure: for return value, see the help for GetStruct structure arguments.

6.18.2 External

class ExternalCls

External commands group definition. 11 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.clone()
```

Subgroups

6.18.2.1 Frequency

SCPI Commands

```
SOURce:EXternal<ExternalGen>:FREQuency
```

class FrequencyCls

Frequency commands group definition. 6 total commands, 4 Subgroups, 1 group commands

get(externalGen=ExternalGen.Nr1) → float

```
# SCPI: SOURce:EXternal<gen>:FREQuency
value: float = driver.source.external.frequency.get(externalGen = repcap.
↳ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

frequency: No help available

set(frequency: float, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURce:EXternal<gen>:FREQuency
driver.source.external.frequency.set(frequency = 1.0, externalGen = repcap.
↳ExternalGen.Nr1)
```

No command help available

param frequency

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.frequency.clone()
```

Subgroups

6.18.2.1.1 Coupling

class CouplingCls

Coupling commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.frequency.coupling.clone()
```

Subgroups

6.18.2.1.1.1 State

SCPI Commands

```
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:COUPLING:STATE
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalGen=ExternalGen.Nr1) → bool

```
# SCPI: SOURCE:EXTERNAL<gen>:FREQUENCY:COUPLING[:STATE]
value: bool = driver.source.external.frequency.coupling.state.get(externalGen = repcap.ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

state: No help available

set(state: bool, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURCE:EXTERNAL<gen>:FREQUENCY:COUPLING[:STATE]
driver.source.external.frequency.coupling.state.set(state = False, externalGen. repcap.ExternalGen.Nr1)
```

No command help available

param state

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.1.2 Factor**class FactorCls**

Factor commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.frequency.factor.clone()
```

Subgroups**6.18.2.1.2.1 Denominator****SCPI Commands**

```
SOURce:EXTernal<ExternalGen>:FREQuency:FACTOR:DENominator
```

class DenominatorCls

Denominator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalGen=ExternalGen.Nr1) → float

```
# SCPI: SOURce:EXTernal<gen>:FREQuency[:FACTOR]:DENominator
value: float = driver.source.external.frequency.factor.denominator.
↳get(externalGen = repcap.ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

value: No help available

set(value: float, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURce:EXTernal<gen>:FREQuency[:FACTOR]:DENominator
driver.source.external.frequency.factor.denominator.set(value = 1.0,↳
↳externalGen = repcap.ExternalGen.Nr1)
```

No command help available

param value

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.1.2.2 Numerator

SCPI Commands

```
SOURce:EXTernal<ExternalGen>:FREQuency:FACTor:NUMerator
```

class NumeratorCls

Numerator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*externalGen=ExternalGen.Nr1*) → float

```
# SCPI: SOURce:EXTernal<gen>:FREQuency[:FACTor]:NUMerator
value: float = driver.source.external.frequency.factor.numerator.
↪get(externalGen = repcap.ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

value: No help available

set(*value: float, externalGen=ExternalGen.Nr1*) → None

```
# SCPI: SOURce:EXTernal<gen>:FREQuency[:FACTor]:NUMerator
driver.source.external.frequency.factor.numerator.set(value = 1.0, externalGen.
↪= repcap.ExternalGen.Nr1)
```

No command help available

param value

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.1.3 Offset

SCPI Commands

```
SOURce:EXTernal<ExternalGen>:FREQuency:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*externalGen=ExternalGen.Nr1*) → float

```
# SCPI: SOURce:EXTernal<gen>:FREQuency:OFFSet
value: float = driver.source.external.frequency.offset.get(externalGen = repcap.
↪ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

offset: No help available

set(offset: float, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURce:EXternal<gen>:FREQuency:OFFSet
driver.source.external.frequency.offset.set(offset = 1.0, externalGen = repcap.
↳ExternalGen.Nr1)
```

No command help available

param offset

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.1.4 Sweep

class SweepCls

Sweep commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.frequency.sweep.clone()
```

Subgroups

6.18.2.1.4.1 State

SCPI Commands

```
SOURce:EXternal<ExternalGen>:FREQuency:SWEep:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalGen=ExternalGen.Nr1) → bool

```
# SCPI: SOURce:EXternal<gen>:FREQuency:SWEep[:STATe]
value: bool = driver.source.external.frequency.sweep.state.get(externalGen =
↳repcap.ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

state: No help available

set(state: bool, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURCE:EXTERNAL<gen>:FREQUENCY:SWEEP[:STATE]
driver.source.external.frequency.sweep.state.set(state = False, externalGen = ↵
↵repcap.ExternalGen.Nr1)
```

No command help available

param state

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.2 Power

class PowerCls

Power commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.power.clone()
```

Subgroups

6.18.2.2.1 Level

SCPI Commands

```
SOURCE:EXTERNAL<ExternalGen>:POWER:LEVEL
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalGen=ExternalGen.Nr1) → float

```
# SCPI: SOURCE:EXTERNAL<gen>:POWER[:LEVEL]
value: float = driver.source.external.power.level.get(externalGen = repcap.
↵ExternalGen.Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

level: No help available

set(level: float, externalGen=ExternalGen.Nr1) → None

```
# SCPI: SOURce:EXternal<gen>:POWer[:LEVel]
driver.source.external.power.level.set(level = 1.0, externalGen = repcap.
↳ExternalGen.Nr1)
```

No command help available

param level

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.2.3 Roscillator

class RoscillatorCls

Roscillator commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.roscillator.clone()
```

Subgroups

6.18.2.3.1 External

class ExternalCls

External commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.roscillator.external.clone()
```

Subgroups

6.18.2.3.1.1 Frequency

SCPI Commands

```
SOURce:EXternal<ExternalRosc>:ROScillator:EXternal:FREQuency
```

class FrequencyCls

Frequency commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(externalRosc=ExternalRosc.Nr1) → float

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator:EXtErnal:FREquency
value: float = driver.source.external.rosillator.external.frequency.
↪get(externalRosc = repcap.ExternalRosc.Nr1)
```

This command defines the frequency of the external reference oscillator. If the external reference oscillator is selected, the reference signal must be connected to the rear panel of the instrument.

param externalRosc

optional repeated capability selector. Default value: Nr1

return

frequency: Range: 1 MHz to 50 MHz, Unit: HZ

set(frequency: float, externalRosc=ExternalRosc.Nr1) → None

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator:EXtErnal:FREquency
driver.source.external.rosillator.external.frequency.set(frequency = 1.0,↪
↪externalRosc = repcap.ExternalRosc.Nr1)
```

This command defines the frequency of the external reference oscillator. If the external reference oscillator is selected, the reference signal must be connected to the rear panel of the instrument.

param frequency

Range: 1 MHz to 50 MHz, Unit: HZ

param externalRosc

optional repeated capability selector. Default value: Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.external.rosillator.external.frequency.clone()
```

Subgroups

6.18.2.3.1.2 Mode

SCPI Commands

```
SOURce:EXtErnal<ExternalRosc>:ROScillator:EXtErnal:FREquency:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalRosc=ExternalRosc.Nr1) → RoscillatorFreqMode

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator:EXtErnal:FREquency:MODE
value: enums.RoscillatorFreqMode = driver.source.external.rosillator.external.
↪frequency.mode.get(externalRosc = repcap.ExternalRosc.Nr1)
```

No command help available

param externalRosc

optional repeated capability selector. Default value: Nr1

return

arg_0: No help available

set(arg_0: *RoscillatorFreqMode*, externalRosc=*ExternalRosc.Nr1*) → None

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator:EXtErnal:FREQuency:MODE
driver.source.external.roscillator.external.frequency.mode.set(arg_0 = enums.
↪RoscillatorFreqMode.E10, externalRosc = repcap.ExternalRosc.Nr1)
```

No command help available

param arg_0

No help available

param externalRosc

optional repeated capability selector. Default value: Nr1

6.18.2.3.2 Source

SCPI Commands

```
SOURce:EXtErnal<ExternalRosc>:ROScillator:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(externalRosc=*ExternalRosc.Nr1*) → ReferenceSourceB

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator[:SOURce]
value: enums.ReferenceSourceB = driver.source.external.roscillator.source.
↪get(externalRosc = repcap.ExternalRosc.Nr1)
```

No command help available

param externalRosc

optional repeated capability selector. Default value: Nr1

return

source: No help available

set(source: *ReferenceSourceB*, externalRosc=*ExternalRosc.Nr1*) → None

```
# SCPI: SOURce:EXtErnal<ext>:ROScillator[:SOURce]
driver.source.external.roscillator.source.set(source = enums.ReferenceSourceB.
↪EAUTO, externalRosc = repcap.ExternalRosc.Nr1)
```

No command help available

param source

No help available

param externalRosc

optional repeated capability selector. Default value: Nr1

6.18.2.4 State

SCPI Commands

```
SOURce:EXtErnal<ExternalGen>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*externalGen=ExternalGen.Nr1*) → bool

```
# SCPI: SOURce:EXtErnal<gen>[:STATe]
value: bool = driver.source.external.state.get(externalGen = repcap.ExternalGen.
↳Nr1)
```

No command help available

param externalGen

optional repeated capability selector. Default value: Nr1

return

state: No help available

set(*state: bool, externalGen=ExternalGen.Nr1*) → None

```
# SCPI: SOURce:EXtErnal<gen>[:STATe]
driver.source.external.state.set(state = False, externalGen = repcap.
↳ExternalGen.Nr1)
```

No command help available

param state

No help available

param externalGen

optional repeated capability selector. Default value: Nr1

6.18.3 Generator

class GeneratorCls

Generator commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.generator.clone()
```

Subgroups

6.18.3.1 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.generator.channel.clone()
```

Subgroups

6.18.3.1.1 Coupling

SCPI Commands

```
SOURce:GENerator:CHANnel:COUpling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:CHANnel:COUpling
value: bool = driver.source.generator.channel.coupling.get()
```

This command couples or decouples the signal source configuration across measurement channels.

return

state: ON | 1 Signal source configuration is the same across all measurement channels.

OFF | 0 Signal source configuration is different for each measurement channel.

set(state: bool) → None

```
# SCPI: SOURce:GENerator:CHANnel:COUpling
driver.source.generator.channel.coupling.set(state = False)
```

This command couples or decouples the signal source configuration across measurement channels.

param state

ON | 1 Signal source configuration is the same across all measurement channels. OFF

| 0 Signal source configuration is different for each measurement channel.

6.18.3.2 DutBypass

SCPI Commands

```
SOURce:GENerator:DUTBypass
```

class DutBypassCls

DutBypass commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:DUTBypass
value: bool = driver.source.generator.dutBypass.get()
```

This command turns the DUT bypass on and off. When you turn on the bypass, the application measures the noise characteristics of the R&S FSWP. The DUT bypass is available with the optional Signal Source hardware component.

return
state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SOURce:GENerator:DUTBypass
driver.source.generator.dutBypass.set(state = False)
```

This command turns the DUT bypass on and off. When you turn on the bypass, the application measures the noise characteristics of the R&S FSWP. The DUT bypass is available with the optional Signal Source hardware component.

param state
ON | OFF | 1 | 0

6.18.3.3 Frequency

SCPI Commands

```
SOURce:GENerator:FREquency
```

class FrequencyCls

Frequency commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:FREquency
value: float = driver.source.generator.frequency.get()
```

This command defines the frequency of the signal that is generated by the signal source.

return
frequency: numeric value Unit: Hz

set(frequency: float) → None

```
# SCPI: SOURce:GENerator:FREQuency
driver.source.generator.frequency.set(frequency = 1.0)
```

This command defines the frequency of the signal that is generated by the signal source.

param frequency
numeric value Unit: Hz

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.generator.frequency.clone()
```

Subgroups

6.18.3.3.1 Step

SCPI Commands

```
SOURce:GENerator:FREQuency:STEP
```

class StepCls

Step commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:FREQuency:STEP
value: float = driver.source.generator.frequency.step.get()
```

This command defines the frequency stepsize of the signal generated by the optional signal source.

return
stepsize: 1 mHz | 1 Hz | 1 kHz | 1 MHz | 1 GHz Unit: Hz

set(stepsize: float) → None

```
# SCPI: SOURce:GENerator:FREQuency:STEP
driver.source.generator.frequency.step.set(stepsize = 1.0)
```

This command defines the frequency stepsize of the signal generated by the optional signal source.

param stepsize
1 mHz | 1 Hz | 1 kHz | 1 MHz | 1 GHz Unit: Hz

6.18.3.4 Level

SCPI Commands

```
SOURce:GENerator:LEVel
```

class LevelCls

Level commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:LEVel
value: float = driver.source.generator.level.get()
```

This command defines the level of the signal that is generated by the signal source.

return

level: numeric value Unit: dBm

set(level: float) → None

```
# SCPI: SOURce:GENerator:LEVel
driver.source.generator.level.set(level = 1.0)
```

This command defines the level of the signal that is generated by the signal source.

param level

numeric value Unit: dBm

6.18.3.5 Modulation

SCPI Commands

```
SOURce:GENerator:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator:MODulation
value: bool = driver.source.generator.modulation.get()
```

This command turns internal pulse modulation for pulsed measurements on and off.

return

state: ON | 1 A pulse is output on the signal source. You can define the pulse characteristics with •method RsFswp.Source.Generator.Pulse.Period.set •method RsFswp.Source.Generator.Pulse.Width.set OFF | 0 A sine signal is output on the signal source.

set(state: bool) → None

```
# SCPI: SOURce:GENerator:MODulation
driver.source.generator.modulation.set(state = False)
```

This command turns internal pulse modulation for pulsed measurements on and off.

param state

ON | 1 A pulse is output on the signal source. You can define the pulse characteristics with •method RsFswp.Source.Generator.Pulse.Period.set •method RsFswp.Source.Generator.Pulse.Width.set OFF | 0 A sine signal is output on the signal source.

6.18.3.6 Pulse

class PulseCls

Pulse commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.generator.pulse.clone()
```

Subgroups

6.18.3.6.1 Period

SCPI Commands

```
SOURce:GENerator:PULSe:PERiod
```

class PeriodCls

Period commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:PULSe:PERiod
value: float = driver.source.generator.pulse.period.get()
```

This command defines the pulse period (distance between two consecutive pulses) of the pulse that is generated.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurements.
- Turn on signal source (method RsFswp.Applications.K30_NoiseFigure.Source.Generator.State.set) .
- Turn on pulse modulation (method RsFswp.Source.Generator.Modulation.set) .

return

pulse_period: numeric value Unit: s

set(pulse_period: float) → None

```
# SCPI: SOURce:GENerator:PULSe:PERiod
driver.source.generator.pulse.period.set(pulse_period = 1.0)
```

This command defines the pulse period (distance between two consecutive pulses) of the pulse that is generated.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurements.
- Turn on signal source (method RsFswp.Applications.K30_NoiseFigure.Source.Generator.State.set) .
- Turn on pulse modulation (method RsFswp.Source.Generator.Modulation.set) .

param pulse_period
numeric value Unit: s

6.18.3.6.2 Trigger

class TriggerCls

Trigger commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.generator.pulse.trigger.clone()
```

Subgroups

6.18.3.6.2.1 Output

SCPI Commands

```
SOURce:GENerator:PULSe:TRIGger:OUTPut
```

class OutputCls

Output commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SignalLevel

```
# SCPI: SOURce:GENerator:PULSe:TRIGger:OUTPut
value: enums.SignalLevel = driver.source.generator.pulse.trigger.output.get()
```

This command selects the signal type provided at the trigger output connector. The signal can be used, for example, to control an external pulse modulator.

return
pulse_output: No help available

set(pulse_output: SignalLevel) → None

```
# SCPI: SOURce:GENerator:PULSe:TRIGger:OUTPut
driver.source.generator.pulse.trigger.output.set(pulse_output = enums.
↪SignalLevel.HIGH)
```

This command selects the signal type provided at the trigger output connector. The signal can be used, for example, to control an external pulse modulator.

param pulse_output

HIGH Provides a high active pulse at the trigger output. Note that the signal is provided even if internal pulse modulation has been turned off. You can define the pulse characteristics with •method RsFswp.Source.Generator.Pulse.Width.set •method RsFswp.Source.Generator.Pulse.Period.set LOW Provides a low active pulse at the trigger output. Note that the signal is provided even if internal pulse modulation has been turned off. OFF | 0 Provides no signal at the trigger output.

6.18.3.6.3 Width

SCPI Commands

```
SOURce:GENerator:PULSe:WIDTh
```

class WidthCls

Width commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:GENerator:PULSe:WIDTh
value: float = driver.source.generator.pulse.width.get()
```

This command defines the length of the pulse that is generated.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurements.
- Turn on signal source (method RsFswp.Applications.K30_NoiseFigure.Source.Generator.State.set)
- Turn on pulse modulation (method RsFswp.Source.Generator.Modulation.set) .

return

pulse_width: numeric value Unit: s

set(pulse_width: float) → None

```
# SCPI: SOURce:GENerator:PULSe:WIDTh
driver.source.generator.pulse.width.set(pulse_width = 1.0)
```

This command defines the length of the pulse that is generated.

INTRO_CMD_HELP: Prerequisites for this command

- Optional pulsed phase noise measurements.
- Turn on signal source (method RsFswp.Applications.K30_NoiseFigure.Source.Generator.State.set)
- Turn on pulse modulation (method RsFswp.Source.Generator.Modulation.set) .

param pulse_width

numeric value Unit: s

6.18.3.7 State

SCPI Commands

SOURce:GENerator:STATe

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:GENerator[:STATe]
value: bool = driver.source.generator.state.get()
```

This command turns the optional signal source output on and off. When you turn on the signal source, the R&S FSWP generates a signal with the frequency and level defined with method RsFswp.Source.Generator.Frequency.set and method RsFswp.Source.Generator.Level.set.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SOURce:GENerator[:STATe]
driver.source.generator.state.set(state = False)
```

This command turns the optional signal source output on and off. When you turn on the signal source, the R&S FSWP generates a signal with the frequency and level defined with method RsFswp.Source.Generator.Frequency.set and method RsFswp.Source.Generator.Level.set.

param state

ON | OFF | 1 | 0

6.18.4 Power

class PowerCls

Power commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.power.clone()
```

Subgroups

6.18.4.1 Level

class LevelCls

Level commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.power.level.clone()
```

Subgroups

6.18.4.1.1 Immediate

class ImmediateCls

Immediate commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.power.level.immediate.clone()
```

Subgroups

6.18.4.1.1.1 Offset

SCPI Commands

```
SOURce:POWer:LEVel:IMMediate:OFFSet
```

class OffsetCls

Offset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:POWer[:LEVel][:IMMediate]:OFFSet
value: float = driver.source.power.level.immediate.offset.get()
```

No command help available

return

offset: No help available

set(offset: float) → None

```
# SCPI: SOURce:POWer[:LEVel][:IMMediate]:OFFSet
driver.source.power.level.immediate.offset.set(offset = 1.0)
```

No command help available

param offset

No help available

6.18.4.2 Sequence

class SequenceCls

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.power.sequence.clone()
```

Subgroups

6.18.4.2.1 Result

SCPI Commands

```
SOURce:POWer:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Power_Vsupply: float: No parameter help available
- Power_Vtune: float: No parameter help available
- Power_Vaux: float: No parameter help available

get() → GetStruct

```
# SCPI: SOURce:POWer:SEquence:RESult
value: GetStruct = driver.source.power.sequence.result.get()
```

This command queries the actually measured power (U*I) on the DC power sources.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)
-

return

structure: for return value, see the help for GetStruct structure arguments.

6.18.5 Temperature

class TemperatureCls

Temperature commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.temperature.clone()
```

Subgroups

6.18.5.1 Frontend

SCPI Commands

```
SOURce:TEMPerature:FRONtend
```

class FrontendCls

Frontend commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:TEMPerature:FRONtend
value: float = driver.source.temperature.frontend.get()
```

This command queries the current frontend temperature of the R&S FSWP. During self-alignment, the instrument's (frontend) temperature is also measured (as soon as the instrument has warmed up completely). This temperature is used as a reference for a continuous temperature check during operation. If the current temperature deviates from the stored self-alignment temperature by a certain degree, a warning is displayed in the status bar indicating the resulting deviation in the measured power levels. A status bit in the STATUS:QUESTIONable:TEMPerature register indicates a possible deviation. (This feature is available in the optional Spectrum and Signal Analyzer application.)

return
temperature: Temperature in degrees Celsius.

6.18.6 Voltage

class VoltageCls

Voltage commands group definition. 17 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.clone()
```

Subgroups

6.18.6.1 Auxiliary

class AuxiliaryCls

Auxiliary commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.auxiliary.clone()
```

Subgroups

6.18.6.1.1 Level

class LevelCls

Level commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.auxiliary.level.clone()
```

Subgroups

6.18.6.1.1.1 Amplitude

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:AMPLitude
value: float = driver.source.voltage.auxiliary.level.amplitude.get()
```

This command defines the output voltage for the Vaux source.

return

voltage: numeric value Range: -10 to 10, Unit: V

set(voltage: float) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:AMPLitude
driver.source.voltage.auxiliary.level.amplitude.set(voltage = 1.0)
```

This command defines the output voltage for the Vaux source.

param voltage

numeric value Range: -10 to 10, Unit: V

6.18.6.1.1.2 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.auxiliary.level.limit.clone()
```

Subgroups

6.18.6.1.1.3 High

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
value: float = driver.source.voltage.auxiliary.level.limit.high.get()
```

This command defines the maximum voltage that may be supplied by the Vaux source.

return

voltage: numeric value Range: -10 to 10, Unit: V

set(voltage: float) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:HIGH
driver.source.voltage.auxiliary.level.limit.high.set(voltage = 1.0)
```

This command defines the maximum voltage that may be supplied by the Vaux source.

param voltage

numeric value Range: -10 to 10, Unit: V

6.18.6.1.1.4 Low

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:LOW
value: float = driver.source.voltage.auxiliary.level.limit.low.get()
```

This command defines the minimum voltage that may be supplied by the Vaux source.

return

voltage: numeric value Range: -10 to 10, Unit: V

set(voltage: float) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel:LIMit:LOW
driver.source.voltage.auxiliary.level.limit.low.set(voltage = 1.0)
```

This command defines the minimum voltage that may be supplied by the Vaux source.

param voltage

numeric value Range: -10 to 10, Unit: V

6.18.6.1.1.5 State

SCPI Commands

```
SOURce:VOLTage:AUX:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:VOLTage:AUX:LEVel[:STATe]
value: bool = driver.source.voltage.auxiliary.level.state.get()
```

This command turns the auxiliary voltage source (Vaux) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)
-

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SOURce:VOLTage:AUX:LEVel[:STATe]
driver.source.voltage.auxiliary.level.state.set(state = False)
```

This command turns the auxiliary voltage source (Vaux) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)
-

param state
ON | OFF | 1 | 0

6.18.6.2 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.channel.clone()
```

Subgroups

6.18.6.2.1 Coupling

SCPI Commands

```
SOURce:VOLTage:CHANnel:COUPling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SOURce:VOLTage:CHANnel:COUPling
value: bool = driver.source.voltage.channel.coupling.get()
```

This command couples or decouples the DC power configuration across measurement channels.

return
state: ON | 1 DC power configuration is the same across all measurement channels.
OFF | 0 DC power configuration is different for each measurement channel.

set(state: bool) → None

```
# SCPI: SOURce:VOLTage:CHANnel:COUPling
driver.source.voltage.channel.coupling.set(state = False)
```

This command couples or decouples the DC power configuration across measurement channels.

param state

ON | 1 DC power configuration is the same across all measurement channels. OFF | 0

DC power configuration is different for each measurement channel.

6.18.6.3 Control<Source>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.source.voltage.control.repcap_source_get()
driver.source.voltage.control.repcap_source_set(repcap.Source.Nr1)
```

class ControlCls

Control commands group definition. 4 total commands, 1 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.control.clone()
```

Subgroups

6.18.6.3.1 Level

class LevelCls

Level commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.control.level.clone()
```

Subgroups

6.18.6.3.1.1 Amplitude

SCPI Commands

```
SOURce:VOLTage:CONTrol<Source>:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:CONTrol<1/2>:LEVel:AMPLitude
value: float = driver.source.voltage.control.level.amplitude.get(source = ↵
↵repcap.Source.Default)
```

This command defines the output voltage for the Vtune source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Control’)

return

voltage: numeric value Range: -10 to 28, Unit: V

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTrol<1/2>:LEVel:AMPLitude
driver.source.voltage.control.level.amplitude.set(voltage = 1.0, source = ↵
↵repcap.Source.Default)
```

This command defines the output voltage for the Vtune source.

param voltage

numeric value Range: -10 to 28, Unit: V

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Control’)

6.18.6.3.1.2 Limit

class LimitCls

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.control.level.limit.clone()
```

Subgroups

6.18.6.3.1.3 High

SCPI Commands

```
SOURce:VOLTage:CONTrol<Source>:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:CONTrol<1/2>:LEVel:LIMit:HIGH
value: float = driver.source.voltage.control.level.limit.high.get(source = ↵
↵repcap.Source.Default)
```

This command defines the maximum voltage that may be supplied by the Vtune source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

voltage: numeric value Range: -10 to 28, Unit: V

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTrol<1/2>:LEVel:LIMit:HIGH
driver.source.voltage.control.level.limit.high.set(voltage = 1.0, source = ↵
↵repcap.Source.Default)
```

This command defines the maximum voltage that may be supplied by the Vtune source.

param voltage

numeric value Range: -10 to 28, Unit: V

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.18.6.3.1.4 Low

SCPI Commands

```
SOURce:VOLTage:CONTrol<Source>:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → float

```
# SCPI: SOURce:VOLTage:CONTrol<1/2>:LEVel:LIMit:LOW
value: float = driver.source.voltage.control.level.limit.low.get(source = ↵
↵repcap.Source.Default)
```

This command defines the minimum voltage that may be supplied by the Vtune source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

voltage: numeric value Range: -10 to 28, Unit: V

set(*voltage: float, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTRol<1/2>:LEVel:LIMit:LOW
driver.source.voltage.control.level.limit.low.set(voltage = 1.0, source = ↵
↵repcap.Source.Default)
```

This command defines the minimum voltage that may be supplied by the Vtune source.

param voltage

numeric value Range: -10 to 28, Unit: V

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.18.6.3.1.5 State

SCPI Commands

```
SOURce:VOLTage:CONTRol<Source>:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → bool

```
# SCPI: SOURce:VOLTage:CONTRol<1/2>:LEVel[:STATe]
value: bool = driver.source.voltage.control.level.state.get(source = repcap.
↵Source.Default)
```

This command turns the tuning voltage source (Vtune) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set).

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

return

state: ON | OFF | 1 | 0

set(*state: bool, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:CONTRol<1/2>:LEVel[:STATe]
driver.source.voltage.control.level.state.set(state = False, source = repcap.
↵Source.Default)
```

This command turns the tuning voltage source (Vtune) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)

param state

ON | OFF | 1 | 0

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Control')

6.18.6.4 Power<Source>**RepCap Settings**

```
# Range: Nr1 .. Nr2
rc = driver.source.voltage.power.repcap_source_get()
driver.source.voltage.power.repcap_source_set(repcap.Source.Nr1)
```

class PowerCls

Power commands group definition. 6 total commands, 2 Subgroups, 0 group commands Repeated Capability: Source, default value after init: Source.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.power.clone()
```

Subgroups**6.18.6.4.1 Level****class LevelCls**

Level commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.power.level.clone()
```

Subgroups**6.18.6.4.1.1 Amplitude****SCPI Commands**

```
SOURce:VOLTage:POWer<Source>:LEVel:AMPLitude
```

class AmplitudeCls

Amplitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel:AMPLitude
value: float = driver.source.voltage.power.level.amplitude.get(source = repcap.
↳Source.Default)
```

This command defines the output voltage for the Vsupply source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage_current: No help available

set(voltage_current: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel:AMPLitude
driver.source.voltage.power.level.amplitude.set(voltage_current = 1.0, source =
↳repcap.Source.Default)
```

This command defines the output voltage for the Vsupply source.

param voltage_current

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.4.1.2 Limit**class LimitCls**

Limit commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.power.level.limit.clone()
```

Subgroups**6.18.6.4.1.3 High****SCPI Commands**

```
SOURce:VOLTage:POWer<Source>:LEVel:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel:LIMit:HIGH
value: float = driver.source.voltage.power.level.limit.high.get(source = repcap.
↳Source.Default)
```

This command defines the maximum voltage that may be supplied by the Vsupply source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage_current: No help available

set(voltage_current: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel:LIMit:HIGH
driver.source.voltage.power.level.limit.high.set(voltage_current = 1.0, source_
↳= repcap.Source.Default)
```

This command defines the maximum voltage that may be supplied by the Vsupply source.

param voltage_current

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.4.1.4 Low**SCPI Commands**

```
SOURce:VOLTage:POWer<Source>:LEVel:LIMit:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel:LIMit:LOW
value: float = driver.source.voltage.power.level.limit.low.get(source = repcap.
↳Source.Default)
```

This command defines the minimum voltage that may be supplied by the Vsupply source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage_current: No help available

set(*voltage_current*: float, *source*=Source.Default) → None

```
# SCPI: SOURCE:VOLTage:POWer<1|2>:LEVel:LIMit:LOW
driver.source.voltage.power.level.limit.low.set(voltage_current = 1.0, source = ↵
↵repcap.Source.Default)
```

This command defines the minimum voltage that may be supplied by the Vsupply source.

param voltage_current

No help available

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.4.1.5 Mode

SCPI Commands

```
SOURCE:VOLTage:POWer<Source>:LEVel:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source*=Source.Default) → PwrLevelMode

```
# SCPI: SOURCE:VOLTage:POWer<1|2>:LEVel:MODE
value: enums.PwrLevelMode = driver.source.voltage.power.level.mode.get(source = ↵
↵repcap.Source.Default)
```

This command selects whether you want to control the output in terms of current or voltage.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

mode: CURRENT Control the output in terms of current. VOLTage Controls the output in terms of voltage.

set(*mode*: PwrLevelMode, *source*=Source.Default) → None

```
# SCPI: SOURCE:VOLTage:POWer<1|2>:LEVel:MODE
driver.source.voltage.power.level.mode.set(mode = enums.PwrLevelMode.CURRENT, ↵
↵source = repcap.Source.Default)
```

This command selects whether you want to control the output in terms of current or voltage.

param mode

CURRENT Control the output in terms of current. VOLTage Controls the output in terms of voltage.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.4.1.6 State

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LEVel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*source=Source.Default*) → bool

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel[:STATe]
value: bool = driver.source.voltage.power.level.state.get(source = repcap.
↳Source.Default)
```

This command turns the supply voltage source (Vsupply) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

state: ON | OFF | 1 | 0

set(*state: bool, source=Source.Default*) → None

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LEVel[:STATe]
driver.source.voltage.power.level.state.set(state = False, source = repcap.
↳Source.Default)
```

This command turns the supply voltage source (Vsupply) on and off. Note that DC power is actually supplied only if you additionally activate the outputs in general.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)

param state

ON | OFF | 1 | 0

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.4.2 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.power.limit.clone()
```

Subgroups

6.18.6.4.2.1 High

SCPI Commands

```
SOURce:VOLTage:POWer<Source>:LIMit:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(source=Source.Default) → float

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LIMit:HIGH
value: float = driver.source.voltage.power.limit.high.get(source = repcap.
↳ Source.Default)
```

This command defines the maximum current or voltage that may be supplied by the Vsupply source.

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

return

voltage: numeric value The type of value depends on whether you control the output in terms of current or voltage (method RsFswp.Source.Voltage.Power.Level.Mode.set) . When you control it in terms of voltage, the value is a current (A) . When you control it in terms of current, the value is a voltage (V) . Range: 0 to 16 V or 2000 mA, Unit: V or A

set(voltage: float, source=Source.Default) → None

```
# SCPI: SOURce:VOLTage:POWer<1|2>:LIMit:HIGH
driver.source.voltage.power.limit.high.set(voltage = 1.0, source = repcap.
↳ Source.Default)
```

This command defines the maximum current or voltage that may be supplied by the Vsupply source.

param voltage

numeric value The type of value depends on whether you control the output in terms of current or voltage (method RsFswp.Source.Voltage.Power.Level.Mode.set) . When you control it in terms of voltage, the value is a current (A) . When you control it in

terms of current, the value is a voltage (V) . Range: 0 to 16 V or 2000 mA, Unit: V or A

param source

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Power')

6.18.6.5 Sequence

class SequenceCls

Sequence commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.voltage.sequence.clone()
```

Subgroups

6.18.6.5.1 Result

SCPI Commands

```
SOURce:VOLTage:SEquence:RESult
```

class ResultCls

Result commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Voltage_Vsupply: float: No parameter help available
- Voltage_Vtune: float: No parameter help available
- Voltage_Vaux: float: No parameter help available

get() → GetStruct

```
# SCPI: SOURce:VOLTage:SEquence:RESult
value: GetStruct = driver.source.voltage.sequence.result.get()
```

This command queries the actually measured voltages on the DC power sources.

INTRO_CMD_HELP: Prerequisites for this command

- Turn on DC power sources (method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.State.set)
-

return

structure: for return value, see the help for GetStruct structure arguments.

6.18.6.6 State

SCPI Commands

SOURce:VOLTage:STATE

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

<pre># SCPI: SOURce:VOLTage[:STATE] value: bool = driver.source.voltage.state.get()</pre>

This command turns DC power sources on and off in general. When you turn off the DC power sources, no power is supplied even when you have turned on one of the sources individually with one of the following commands.

INTRO_CMD_HELP: Prerequisites for this command

- method RsFswp.Source.Voltage.Auxiliary.Level.State.set
- method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Control.Level.State.set
- method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Power.Level.State.set

Note that you can turn on the global power supply if at least one of the individual supplies has been turned on.

return

state: ON | 1 DC power sources are ready for use. OFF | 0 DC power sources are turned off.

set(state: bool) → None

<pre># SCPI: SOURce:VOLTage[:STATE] driver.source.voltage.state.set(state = False)</pre>
--

This command turns DC power sources on and off in general. When you turn off the DC power sources, no power is supplied even when you have turned on one of the sources individually with one of the following commands.

INTRO_CMD_HELP: Prerequisites for this command

- method RsFswp.Source.Voltage.Auxiliary.Level.State.set
- method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Control.Level.State.set
- method RsFswp.Applications.K30_NoiseFigure.Source.Voltage.Power.Level.State.set

Note that you can turn on the global power supply if at least one of the individual supplies has been turned on.

param state

ON | 1 DC power sources are ready for use. OFF | 0 DC power sources are turned off.

6.19 Status

SCPI Commands

STATus:PRESet

class StatusCls

Status commands group definition. 112 total commands, 3 Subgroups, 1 group commands

preset() → None

<pre># SCPI: STATus:PRESet driver.status.preset()</pre>

This command resets the edge detectors and ENABLE parts of all registers to a defined value. All PTRansition parts are set to FFFFh, i.e. all transitions from 0 to 1 are detected. All NTRansition parts are set to 0, i.e. a transition from 1 to 0 in a CONDition bit is not detected. The ENABLE part of the method **RsFswp.Status.Operation.Event.get_** and method **RsFswp.Status.Questionable.Event.get_** registers are set to 0, i.e. all events in these registers are not passed on.

preset_with_opc(opc_timeout_ms: int = -1) → None

<pre># SCPI: STATus:PRESet driver.status.preset_with_opc()</pre>
--

This command resets the edge detectors and ENABLE parts of all registers to a defined value. All PTRansition parts are set to FFFFh, i.e. all transitions from 0 to 1 are detected. All NTRansition parts are set to 0, i.e. a transition from 1 to 0 in a CONDition bit is not detected. The ENABLE part of the method **RsFswp.Status.Operation.Event.get_** and method **RsFswp.Status.Questionable.Event.get_** registers are set to 0, i.e. all events in these registers are not passed on.

Same as preset, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

<pre># Create a clone of the original group, that exists independently group2 = driver.status.clone()</pre>

Subgroups

6.19.1 Operation

class OperationCls

Operation commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.operation.clone()
```

Subgroups

6.19.1.1 Condition

SCPI Commands

```
STATus:OPERation:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATus:OPERation:CONDition
value: int = driver.status.operation.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return
register_value: No help available

6.19.1.2 Enable

SCPI Commands

```
STATus:OPERation:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATus:OPERation:ENABle
value: int = driver.status.operation.enable.get()
```

These commands control the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return
summary_bit: No help available

set(summary_bit: int) → None

```
# SCPI: STATus:OPERation:ENABle
driver.status.operation.enable.set(summary_bit = 1)
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

6.19.1.3 Event

SCPI Commands

```
STATUS:OPERation:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATUS:OPERation[:EVENT]
value: int = driver.status.operation.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

register_value: No help available

6.19.1.4 Ntransition

SCPI Commands

```
STATUS:OPERation:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATUS:OPERation:NTRansition
value: int = driver.status.operation.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

summary_bit: No help available

set(summary_bit: int) → None

```
# SCPI: STATUS:OPERation:NTRansition
driver.status.operation.ntransition.set(summary_bit = 1)
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

6.19.1.5 Pcalibration

class PcalibrationCls

Pcalibration commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.operation.pcalibration.clone()
```

Subgroups

6.19.1.5.1 Condition

SCPI Commands

```
STATUS:OPERation:PCALibration:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:OPERation:PCALibration:CONDition
value: str = driver.status.operation.pcalibration.condition.get()
```

No command help available

return

channel_name: No help available

6.19.1.5.2 Enable

SCPI Commands

```
STATUS:OPERation:PCALibration:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATus:OPERation:PCALibration:ENABLE
value: EnableStruct = driver.status.operation.pcalibration.enable.get()
```

No command help available

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:OPERation:PCALibration:ENABLE
driver.status.operation.pcalibration.enable.set(summary_bit = 1, channel_name =
↳ '1')
```

No command help available

param summary_bit

No help available

param channel_name

No help available

6.19.1.5.3 Event**SCPI Commands**

```
STATus:OPERation:PCALibration:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:OPERation:PCALibration[:EVENT]
value: str = driver.status.operation.pcalibration.event.get()
```

No command help available

return

channel_name: No help available

6.19.1.5.4 Ntransission

SCPI Commands

STATus:OPERation:PCALibration:NTRansission

class NtransissionCls

Ntransission commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransissionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransissionStruct

```
# SCPI: STATus:OPERation:PCALibration:NTRansission
value: NtransissionStruct = driver.status.operation.pcalibration.ntransission.
↳get()
```

No command help available

return

structure: for return value, see the help for NtransissionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:OPERation:PCALibration:NTRansission
driver.status.operation.pcalibration.ntransission.set(summary_bit = 1, channel_
↳name = '1')
```

No command help available

param summary_bit

No help available

param channel_name

No help available

6.19.1.5.5 Ptransission

SCPI Commands

STATus:OPERation:PCALibration:PTRansission

class PtransissionCls

Ptransission commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransissionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransistionStruct

```
# SCPI: STATus:OPERation:PCALibration:PTRansistion
value: PtransistionStruct = driver.status.operation.pcalibration.ptransistion.
↳get()
```

No command help available

return

structure: for return value, see the help for PtransistionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:OPERation:PCALibration:PTRansistion
driver.status.operation.pcalibration.ptransistion.set(summary_bit = 1, channel_
↳name = '1')
```

No command help available

param summary_bit

No help available

param channel_name

No help available

6.19.1.6 Ptransition

SCPI Commands

STATus:OPERation:PTRansition

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATus:OPERation:PTRansition
value: int = driver.status.operation.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

summary_bit: No help available

set(summary_bit: int) → None

```
# SCPI: STATus:OPERation:PTRansition
driver.status.operation.ptransition.set(summary_bit = 1)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

6.19.2 Questionable

class QuestionableCls

Questionable commands group definition. 100 total commands, 20 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.clone()
```

Subgroups

6.19.2.1 AcpLimit

class AcpLimitCls

AcpLimit commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.acpLimit.clone()
```

Subgroups

6.19.2.1.1 Condition

SCPI Commands

```
STATus:QUEStionable:ACPLimit:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:ACPLimit:CONDition
value: str = driver.status.questionable.acpLimit.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.1.2 Enable

SCPI Commands

```
STATus:QUESTionable:ACPLimit:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUESTionable:ACPLimit:ENABLE
value: EnableStruct = driver.status.questionable.acpLimit.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:ACPLimit:ENABLE
driver.status.questionable.acpLimit.enable.set(summary_bit = 1, channel_name =
↳ '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.1.3 Event

SCPI Commands

```
STATus:QUEStionable:ACPLimit:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:ACPLimit[:EVENT]
value: str = driver.status.questionable.acpLimit.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.1.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:ACPLimit:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:ACPLimit:NTRansition
value: NtransitionStruct = driver.status.questionable.acpLimit.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:ACPLimit:NTRansition
driver.status.questionable.acpLimit.ntransition.set(summary_bit = 1, channel_
↳ name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.1.5 Ptransition

SCPI Commands

```
STATUS:QUESTionable:ACPLimit:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATUS:QUESTionable:ACPLimit:PTRansition
value: PtransitionStruct = driver.status.questionable.acpLimit.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTionable:ACPLimit:PTRansition
driver.status.questionable.acpLimit.ptransition.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.2 Calibration

class CalibrationCls

Calibration commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.calibration.clone()
```

Subgroups

6.19.2.2.1 Condition

SCPI Commands

```
STATUS:QUESTIONable:CALibration:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONable:CALibration:CONDition
value: str = driver.status.questionable.calibration.condition.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.2.2 Enable

SCPI Commands

```
STATUS:QUESTIONable:CALibration:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATUS:QUESTIONable:CALibration:ENABle
value: EnableStruct = driver.status.questionable.calibration.enable.get()
```

No command help available

return

structure: for return value, see the help for EnableStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:CALibration:ENABle
driver.status.questionable.calibration.enable.set(bit_definition = 1, channel_
↳ name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.2.3 Event

SCPI Commands

```
STATus:QUEStionable:CALibration:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:CALibration[:EVENT]
value: str = driver.status.questionable.calibration.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.2.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:CALibration:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:CALibration:NTRansition
value: NtransitionStruct = driver.status.questionable.calibration.ntransition.
↳get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATus:QUEStionable:CALibration:NTRansition
driver.status.questionable.calibration.ntransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.2.5 Ptransition

SCPI Commands

STATus:QUEStionable:CALibration:PTRansition

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:CALibration:PTRansition
value: PtransitionStruct = driver.status.questionable.calibration.pttransition.
↳get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATus:QUEStionable:CALibration:PTRansition
driver.status.questionable.calibration.pttransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.3 Condition

SCPI Commands

```
STATUS:QUESTIONable:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATUS:QUESTIONable:CONDition
value: int = driver.status.questionable.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

register_value: No help available

6.19.2.4 Correction

class CorrectionCls

Correction commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.correction.clone()
```

Subgroups

6.19.2.4.1 Condition

SCPI Commands

```
STATUS:QUESTIONable:CORRection:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:CORRection:CONDition
value: str = driver.status.questionable.correction.condition.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.4.2 Enable

SCPI Commands

```
STATus:QUESTionable:CORRection:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATus:QUESTionable:CORRection:ENABLE
value: EnableStruct = driver.status.questionable.correction.enable.get()
```

No command help available

```
return
    structure: for return value, see the help for EnableStruct structure arguments.
```

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:CORRection:ENABLE
driver.status.questionable.correction.enable.set(summary_bit = 1, channel_name_
↪ = '1')
```

No command help available

```
param summary_bit
    No help available

param channel_name
    No help available
```


6.19.2.4.3 Event

SCPI Commands

```
STATUS:QUESTIONABLE:CORRection:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONABLE:CORRection[:EVENT]
value: str = driver.status.questionable.correction.event.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.4.4 Ntransition

SCPI Commands

```
STATUS:QUESTIONABLE:CORRection:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:CORRection:NTRansition
value: NtransitionStruct = driver.status.questionable.correction.ntransition.
    ↪get()
```

No command help available

```
return
    structure: for return value, see the help for NtransitionStruct structure arguments.
```

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTIONABLE:CORRection:NTRansition
driver.status.questionable.correction.ntransition.set(summary_bit = 1, channel_
    ↪name = '1')
```

No command help available

```
param summary_bit
    No help available
```

param channel_name
No help available

6.19.2.4.5 Ptransition

SCPI Commands

`STATUS:QUESTionable:CORRection:PTRansition`

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATUS:QUESTionable:CORRection:PTRansition
value: PtransitionStruct = driver.status.questionable.correction.ptransition.
↳get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTionable:CORRection:PTRansition
driver.status.questionable.correction.ptransition.set(summary_bit = 1, channel_
↳name = '1')
```

No command help available

param summary_bit
No help available

param channel_name
No help available

6.19.2.5 Diq

class DiqCls

Diq commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.diq.clone()
```

Subgroups

6.19.2.5.1 Condition

SCPI Commands

```
STATus:QUEStionable:DIQ:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:DIQ:CONDition
value: str = driver.status.questionable.diq.condition.get()
```

No command help available

```
return
channel_name: No help available
```

6.19.2.5.2 Enable

SCPI Commands

```
STATus:QUEStionable:DIQ:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATus:QUEStionable:DIQ:ENABLE
value: EnableStruct = driver.status.questionable.diq.enable.get()
```

No command help available

```
return
structure: for return value, see the help for EnableStruct structure arguments.
```

set(*bit_definition*: int, *channel_name*: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:DIQ:ENABle
driver.status.questionable.diq.enable.set(bit_definition = 1, channel_name = '1
↪')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.5.3 Event

SCPI Commands

```
STATus:QUEStionable:DIQ:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:DIQ[:EVENT]
value: str = driver.status.questionable.diq.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.5.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:DIQ:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:DIQ:NTRansition
value: NtransitionStruct = driver.status.questionable.diq.ntransition.get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATus:QUEStionable:DIQ:NTRansition
driver.status.questionable.diq.ntransition.set(bit_definition = 1, channel_name_
↳= '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.5.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:DIQ:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:DIQ:PTRansition
value: PtransitionStruct = driver.status.questionable.diq.ptransition.get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATus:QUEStionable:DIQ:PTRansition
driver.status.questionable.diq.ptransition.set(bit_definition = 1, channel_name_
↳= '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.6 Enable

SCPI Commands

STATus:QUESTionable:ENABle

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATus:QUESTionable:ENABle
value: int = driver.status.questionable.enable.get()
```

These commands control the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

summary_bit: No help available

set(summary_bit: int) → None

```
# SCPI: STATus:QUESTionable:ENABle
driver.status.questionable.enable.set(summary_bit = 1)
```

These commands control the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

6.19.2.7 Event

SCPI Commands

STATus:QUESTionable:EVENT

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATus:QUESTionable[:EVENT]
value: int = driver.status.questionable.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

register_value: No help available

6.19.2.8 Extended

class ExtendedCls

Extended commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.extended.clone()
```

Subgroups

6.19.2.8.1 Condition

SCPI Commands

```
STATUS:QUESTIONable:EXTended:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONable:EXTended:CONDition
value: str = driver.status.questionable.extended.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.8.2 Enable

SCPI Commands

```
STATUS:QUESTIONable:EXTended:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUESTionable:EXTended:ENABLE
value: EnableStruct = driver.status.questionable.extended.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:EXTended:ENABLE
driver.status.questionable.extended.enable.set(summary_bit = 1, channel_name =
↳ '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.3 Event

SCPI Commands

```
STATus:QUESTionable:EXTended:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:EXTended[:EVENT]
value: str = driver.status.questionable.extended.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.4 Info

class InfoCls

Info commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.extended.info.clone()
```

Subgroups

6.19.2.8.4.1 Condition

SCPI Commands

```
STATUS:QUESTionable:EXTended:INFO:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTionable:EXTended:INFO:CONDition
value: str = driver.status.questionable.extended.info.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.8.4.2 Enable

SCPI Commands

```
STATUS:QUESTionable:EXTended:INFO:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUEStionable:EXTended:INFO:ENABLE
value: EnableStruct = driver.status.questionable.extended.info.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:EXTended:INFO:ENABLE
driver.status.questionable.extended.info.enable.set(summary_bit = 1, channel_
↪ name = '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.4.3 Event

SCPI Commands

```
STATus:QUEStionable:EXTended:INFO:EVENTt
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:EXTended:INFO[:EVENTt]
value: str = driver.status.questionable.extended.info.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.4.4 Ntransition

SCPI Commands

```
STATus:QUESTionable:EXTended:INFO:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUESTionable:EXTended:INFO:NTRansition
value: NtransitionStruct = driver.status.questionable.extended.info.ntransition.
↳get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:EXTended:INFO:NTRansition
driver.status.questionable.extended.info.ntransition.set(summary_bit = 1,
↳channel_name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.4.5 Ptransition

SCPI Commands

`STATUS:QUESTIONable:EXTended:INFO:PTRansition`

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATUS:QUESTIONable:EXTended:INFO:PTRansition
value: PtransitionStruct = driver.status.questionable.extended.info.ptransition.
↳get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTIONable:EXTended:INFO:PTRansition
driver.status.questionable.extended.info.ptransition.set(summary_bit = 1,
↳channel_name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.5 Ntransition

SCPI Commands

```
STATus:QUEStionable:EXTended:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:EXTended:NTRansition
value: NtransitionStruct = driver.status.questionable.extended.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:EXTended:NTRansition
driver.status.questionable.extended.ntransition.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.8.6 Ptransition

SCPI Commands

```
STATus:QUEStionable:EXTended:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:EXTended:PTRansition
value: PtransitionStruct = driver.status.questionable.extended.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:EXTended:PTRansition
driver.status.questionable.extended.ptransition.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.9 Frequency

class FrequencyCls

Frequency commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.frequency.clone()
```

Subgroups

6.19.2.9.1 Condition

SCPI Commands

```
STATus:QUESTionable:FREQuency:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:FREQuency:CONDition
value: str = driver.status.questionable.frequency.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.9.2 Enable

SCPI Commands

```
STATus:QUESTionable:FREQuency:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUESTionable:FREQuency:ENABle
value: EnableStruct = driver.status.questionable.frequency.enable.get()
```

These commands control the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:FREQuency:ENABle
driver.status.questionable.frequency.enable.set(summary_bit = 1, channel_name =
↪ '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.9.3 Event

SCPI Commands

```
STATus:QUESTionable:FREQuency:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:FREQuency[:EVENT]
value: str = driver.status.questionable.frequency.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.9.4 Ntransition

SCPI Commands

```
STATus:QUESTionable:FREQuency:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available

- **Channel_Name:** str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:FREQuency:NTRansition
value: NtransitionStruct = driver.status.questionable.frequency.ntransition.
↪get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:FREQuency:NTRansition
driver.status.questionable.frequency.ntransition.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.9.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:FREQuency:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit:** int: No parameter help available
- **Channel_Name:** str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:FREQuency:PTRansition
value: PtransitionStruct = driver.status.questionable.frequency.ptransition.
↪get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:FREQuency:PTRansition
driver.status.questionable.frequency.ptransition.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.10 Integrity

class IntegrityCls

Integrity commands group definition. 15 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.integrity.clone()
```

Subgroups

6.19.2.10.1 Condition

SCPI Commands

```
STATus:QUEStionable:INTEgrity:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:INTEgrity:CONDition
value: str = driver.status.questionable.integrity.condition.get()
```

No command help available

return
channel_name: No help available

6.19.2.10.2 Enable

SCPI Commands

```
STATUS:QUESTionable:INTEgrity:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATUS:QUESTionable:INTEgrity:ENABle
value: EnableStruct = driver.status.questionable.integrity.enable.get()
```

No command help available

return
structure: for return value, see the help for EnableStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTionable:INTEgrity:ENABle
driver.status.questionable.integrity.enable.set(bit_definition = 1, channel_
name = '1')
```

No command help available

param bit_definition
No help available

param channel_name
No help available

6.19.2.10.3 Event

SCPI Commands

```
STATUS:QUESTionable:INTEgrity:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:INTEgrity[:EVENT]
value: str = driver.status.questionable.integrity.event.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.10.4 Ntransition

SCPI Commands

```
STATus:QUESTionable:INTEgrity:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATus:QUESTionable:INTEgrity:NTRansition
value: NtransitionStruct = driver.status.questionable.integrity.ntransition.
↳get()
```

No command help available

```
return
    structure: for return value, see the help for NtransitionStruct structure arguments.
```

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:INTEgrity:NTRansition
driver.status.questionable.integrity.ntransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

```
param bit_definition
    No help available

param channel_name
    No help available
```

6.19.2.10.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:INTegrity:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:INTegrity:PTRansition
value: PtransitionStruct = driver.status.questionable.integrity.ptransition.
↳get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:INTegrity:PTRansition
driver.status.questionable.integrity.ptransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.10.6 Signal

class SignalCls

Signal commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.integrity.signal.clone()
```

Subgroups

6.19.2.10.6.1 Condition

SCPI Commands

```
STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:CONDITION
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:CONDITION
value: str = driver.status.questionable.integrity.signal.condition.get()
```

No command help available

return
channel_name: No help available

6.19.2.10.6.2 Enable

SCPI Commands

```
STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:ENABLE
value: EnableStruct = driver.status.questionable.integrity.signal.enable.get()
```

No command help available

return
structure: for return value, see the help for EnableStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:ENABLE
driver.status.questionable.integrity.signal.enable.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.10.6.3 Event

SCPI Commands

```
STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL[:EVENT]
value: str = driver.status.questionable.integrity.signal.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.10.6.4 Ntransition

SCPI Commands

```
STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:NTRANSITION
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:NTRANSITION
value: NtransitionStruct = driver.status.questionable.integrity.signal.
↳ntransition.get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:NTRANSITION
driver.status.questionable.integrity.signal.ntransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.10.6.5 Ptransition

SCPI Commands

```
STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:PTRANSITION
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:PTRANSITION
value: PtransitionStruct = driver.status.questionable.integrity.signal.
↳ptransition.get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATUS:QUESTIONABLE:INTEGRITY:SIGNAL:PTRANSITION
driver.status.questionable.integrity.signal.ptransition.set(bit_definition = 1,
↳channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name
No help available

6.19.2.10.7 Uncalibrated

class UncalibratedCls

Uncalibrated commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.integrity.uncalibrated.clone()
```

Subgroups

6.19.2.10.7.1 Condition

SCPI Commands

```
STATUS:QUESTionable:INTEgrity:UNCalibrated:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTionable:INTEgrity:UNCalibrated:CONDition
value: str = driver.status.questionable.integrity.uncalibrated.condition.get()
```

No command help available

return
channel_name: No help available

6.19.2.10.7.2 Enable

SCPI Commands

```
STATUS:QUESTionable:INTEgrity:UNCalibrated:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated:ENABLE
value: EnableStruct = driver.status.questionable.integrity.uncalibrated.enable.
↪ get()
```

No command help available

return

structure: for return value, see the help for EnableStruct structure arguments.

set(*bit_definition*: int, *channel_name*: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated:ENABLE
driver.status.questionable.integrity.uncalibrated.enable.set(bit_definition = 1,
↪ channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.10.7.3 Event

SCPI Commands

```
STATus:QUEStionable:INTEgrity:UNCalibrated:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated[:EVENT]
value: str = driver.status.questionable.integrity.uncalibrated.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.10.7.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:INTEgrity:UNCalibrated:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated:NTRansition
value: NtransitionStruct = driver.status.questionable.integrity.uncalibrated.
↳ ntransition.get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated:NTRansition
driver.status.questionable.integrity.uncalibrated.ntransition.set(bit_
↳ definition = 1, channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.10.7.5 Ptransition**SCPI Commands**

STATus:QUEStionable:INTEgrity:UNCalibrated:PTRansition

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:INTEgrity:UNCalibrated:PTRansition
value: PtransitionStruct = driver.status.questionable.integrity.uncalibrated.
↳ ptransition.get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

`set(bit_definition: int, channel_name: Optional[str] = None) → None`

```
# SCPI: STATus:QUEStionable:INTEGRity:UNCalibrated:PTRansition
driver.status.questionable.integrity.uncalibrated.ptransition.set(bit_
↪definition = 1, channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.11 Limit<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.status.questionable.limit.repcap_window_get()
driver.status.questionable.limit.repcap_window_set(repcap.Window.Nr1)
```

class LimitCls

Limit commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.limit.clone()
```

Subgroups

6.19.2.11.1 Condition

SCPI Commands

```
STATus:QUEStionable:LIMit<Window>:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`get(window=Window.Default) → str`

```
# SCPI: STATus:QUEStionable:LIMit<1/2/3/4>:CONDition
value: str = driver.status.questionable.limit.condition.get(window = repcap.
↪Window.Default)
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.11.2 Enable**SCPI Commands**

```
STaTus:QUEStionable:LIMit<Window>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=*Window.Default*) → EnableStruct

```
# SCPI: STaTus:QUEStionable:LIMit<1|2|3|4>:ENABle
value: EnableStruct = driver.status.questionable.limit.enable.get(window = ↪
↪repcap.Window.Default)
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=*Window.Default*) → None

```
# SCPI: STaTus:QUEStionable:LIMit<1|2|3|4>:ENABle
driver.status.questionable.limit.enable.set(summary_bit = 1, channel_name = '1',
↪ window = repcap.Window.Default)
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.19.2.11.3 Event

SCPI Commands

STATUS:QUESTIONable:LIMit<Window>:EVENTt

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=*Window.Default*) → str

```
# SCPI: STATUS:QUESTIONable:LIMit<1|2|3|4>[:EVENTt]
value: str = driver.status.questionable.limit.event.get(window = repcap.Window.
↳Default)
```

These commands read out the EVENTt section of the status register. At the same time, the commands delete the contents of the EVENTt section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.11.4 Ntransition

SCPI Commands

STATUS:QUESTIONable:LIMit<Window>:NTRansition

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=*Window.Default*) → NtransitionStruct

```
# SCPI: STATus:QUEStionable:LIMit<1/2/3/4>:NTRansition
value: NtransitionStruct = driver.status.questionable.limit.ntransition.
↳ get(window = repcap.Window.Default)
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:LIMit<1/2/3/4>:NTRansition
driver.status.questionable.limit.ntransition.set(summary_bit = 1, channel_name_
↳ = '1', window = repcap.Window.Default)
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.19.2.11.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:LIMit<Window>:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → PtransitionStruct

```
# SCPI: STATus:QUEStionable:LIMit<1/2/3/4>:PTRansition
value: PtransitionStruct = driver.status.questionable.limit.ptransition.
↳ get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUEStionable:LIMit<1/2/3/4>:PTRansition
driver.status.questionable.limit.ptransition.set(summary_bit = 1, channel_name_
↳ = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Limit')

6.19.2.12 Lmargin<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.status.questionable.lmargin.repcap_window_get()
driver.status.questionable.lmargin.repcap_window_set(repcap.Window.Nr1)
```

class LmarginCls

Lmargin commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.lmargin.clone()
```

Subgroups

6.19.2.12.1 Condition

SCPI Commands

```
STATus:QUEStionable:LMARgin<Window>:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUEStionable:LMARgin<1|2|3|4>:CONDition
value: str = driver.status.questionable.lmargin.condition.get(window = repcap.
↳Window.Default)
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.12.2 Enable

SCPI Commands

```
STATus:QUEStionable:LMARgin<Window>:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → EnableStruct

```
# SCPI: STATus:QUESTionable:LMARgin<1/2/3/4>:ENABLE
value: EnableStruct = driver.status.questionable.lmargin.enable.get(window =
↳ repcap.Window.Default)
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=Window.Default) → None

```
# SCPI: STATus:QUESTionable:LMARgin<1/2/3/4>:ENABLE
driver.status.questionable.lmargin.enable.set(summary_bit = 1, channel_name = '1
↳ ', window = repcap.Window.Default)
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

6.19.2.12.3 Event

SCPI Commands

```
STATus:QUESTionable:LMARgin<Window>:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → str

```
# SCPI: STATus:QUESTionable:LMARgin<1/2/3/4>[:EVENT]
value: str = driver.status.questionable.lmargin.event.get(window = repcap.
↳ Window.Default)
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.12.4 Ntransition**SCPI Commands**

STATUS:QUESTionable:LMARgin<Window>:NTRansition

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=Window.Default) → NtransitionStruct

<pre># SCPI: STATUS:QUESTionable:LMARgin<1 2 3 4>:NTRansition value: NtransitionStruct = driver.status.questionable.lmargin.ntransition. ↳get(window = repcap.Window.Default)</pre>

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=Window.Default) → None

<pre># SCPI: STATUS:QUESTionable:LMARgin<1 2 3 4>:NTRansition driver.status.questionable.lmargin.ntransition.set(summary_bit = 1, channel_ ↳name = '1', window = repcap.Window.Default)</pre>

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

6.19.2.12.5 Ptransition**SCPI Commands**

```
STaTus:QUEStionable:LMARgin<Window>:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get(window=*Window.Default*) → PtransitionStruct

```
# SCPI: STaTus:QUEStionable:LMARgin<1|2|3|4>:PTRansition
value: PtransitionStruct = driver.status.questionable.lmargin.ptransition.
↪get(window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVEnt register.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None, window=*Window.Default*) → None

```
# SCPI: STaTus:QUEStionable:LMARgin<1|2|3|4>:PTRansition
driver.status.questionable.lmargin.ptransition.set(summary_bit = 1, channel_
↪name = '1', window = repcap.Window.Default)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVEnt register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Lmargin')

6.19.2.13 Ntransition**SCPI Commands**

STATUS:QUESTIONable:NTRansition

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

<pre># SCPI: STATUS:QUESTIONable:NTRansition value: int = driver.status.questionable.ntransition.get()</pre>
--

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

summary_bit: No help available

set(summary_bit: int) → None

<pre># SCPI: STATUS:QUESTIONable:NTRansition driver.status.questionable.ntransition.set(summary_bit = 1)</pre>
--

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

6.19.2.14 Pnoise**class PnoiseCls**

Pnoise commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.pnoise.clone()
```

Subgroups

6.19.2.14.1 Condition

SCPI Commands

```
STATUS:QUESTIONable:PNOise:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONable:PNOise:CONDition
value: str = driver.status.questionable.pnoise.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.14.2 Enable

SCPI Commands

```
STATUS:QUESTIONable:PNOise:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATUS:QUESTIONable:PNOise:ENABle
value: EnableStruct = driver.status.questionable.pnoise.enable.get()
```

These commands control the ENABle part of a register. The ENABle part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and

its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:PNOise:ENABle
driver.status.questionable.pnoise.enable.set(summary_bit = 1, channel_name = '1
↪')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.14.3 Event

SCPI Commands

```
STATus:QUEStionable:PNOise:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:PNOise[:EVENT]
value: str = driver.status.questionable.pnoise.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.14.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:PNOise:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:PNOise:NTRansition
value: NtransitionStruct = driver.status.questionable.pnoise.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:PNOise:NTRansition
driver.status.questionable.pnoise.ntransition.set(summary_bit = 1, channel_name_
↪= '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.14.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:PNOise:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct


```
# SCPI: STATus:QUEStionable:PNOise:PTRansition
value: PtransitionStruct = driver.status.questionable.pnoise.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:PNOise:PTRansition
driver.status.questionable.pnoise.ptransition.set(summary_bit = 1, channel_name_
↳= '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15 Power

class PowerCls

Power commands group definition. 10 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.power.clone()
```

Subgroups

6.19.2.15.1 Condition

SCPI Commands

```
STATus:QUEStionable:POWer:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:POWer:CONDition
value: str = driver.status.questionable.power.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.15.2 DcpNoise

class DcpNoiseCls

DcpNoise commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.power.dcpNoise.clone()
```

Subgroups

6.19.2.15.2.1 Condition

SCPI Commands

```
STATus:QUEStionable:POWer:DCPNoise:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:POWer:DCPNoise:CONDition
value: str = driver.status.questionable.power.dcpNoise.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.15.2.2 Enable

SCPI Commands

```
STATus:QUEStionable:POWer:DCPNoise:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUEStionable:POWer:DCPNoise:ENABle
value: EnableStruct = driver.status.questionable.power.dcpNoise.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:POWer:DCPNoise:ENABle
driver.status.questionable.power.dcpNoise.enable.set(summary_bit = 1, channel_
↪name = '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15.2.3 Event**SCPI Commands**

```
STATus:QUEStionable:POWer:DCPNoise:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:POWer:DCPNoise[:EVENT]
value: str = driver.status.questionable.power.dcpNoise.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.15.2.4 Ntransition**SCPI Commands**

STATUS:QUEStionable:POWer:DCPNoise:NTRansition

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATUS:QUEStionable:POWer:DCPNoise:NTRansition
value: NtransitionStruct = driver.status.questionable.power.dcpNoise.
↳ ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUEStionable:POWer:DCPNoise:NTRansition
driver.status.questionable.power.dcpNoise.ntransition.set(summary_bit = 1,
↳ channel_name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15.2.5 Ptransition

SCPI Commands

```
STATus:QUESTionable:POWer:DCPNoise:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATus:QUESTionable:POWer:DCPNoise:PTRansition
value: PtransitionStruct = driver.status.questionable.power.dcpNoise.
↳ ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:POWer:DCPNoise:PTRansition
driver.status.questionable.power.dcpNoise.ptransition.set(summary_bit = 1,
↳ channel_name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15.3 Enable

SCPI Commands

`STATus:QUESTionable:POWer:ENABle`

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- `Summary_Bit`: int: No parameter help available
- `Channel_Name`: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

`get()` → `EnableStruct`

```
# SCPI: STATus:QUESTionable:POWer:ENABle
value: EnableStruct = driver.status.questionable.power.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for `EnableStruct` structure arguments.

`set(summary_bit: int, channel_name: Optional[str] = None)` → None

```
# SCPI: STATus:QUESTionable:POWer:ENABle
driver.status.questionable.power.enable.set(summary_bit = 1, channel_name = '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to bereported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15.4 Event

SCPI Commands

```
STATus:QUESTionable:POWer:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:POWer[:EVENT]
value: str = driver.status.questionable.power.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.15.5 Ntransition

SCPI Commands

```
STATus:QUESTionable:POWer:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUESTionable:POWer:NTRansition
value: NtransitionStruct = driver.status.questionable.power.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:POWer:NTRansition
driver.status.questionable.power.ntransition.set(summary_bit = 1, channel_name_
↳ = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.15.6 Ptransition

SCPI Commands

STATus:QUEStionable:POWer:PTRansition

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

<pre># SCPI: STATus:QUEStionable:POWer:PTRansition value: PtransitionStruct = driver.status.questionable.power.ptransition.get()</pre>
--

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

<pre># SCPI: STATus:QUEStionable:POWer:PTRansition driver.status.questionable.power.ptransition.set(summary_bit = 1, channel_name_ ↪= '1')</pre>
--

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.16 Ptransition

SCPI Commands

```
STATUS:QUESTionable:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → int

```
# SCPI: STATUS:QUESTionable:PTRansition
value: int = driver.status.questionable.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

summary_bit: No help available

set(summary_bit: int) → None

```
# SCPI: STATUS:QUESTionable:PTRansition
driver.status.questionable.ptransition.set(summary_bit = 1)
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

6.19.2.17 Sync

class SyncCls

Sync commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.sync.clone()
```

Subgroups

6.19.2.17.1 Condition

SCPI Commands

STATUS:QUESTionable:SYNC:CONDition

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTionable:SYNC:CONDition
value: str = driver.status.questionable.sync.condition.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.17.2 Enable

SCPI Commands

STATUS:QUESTionable:SYNC:ENABle

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATUS:QUESTionable:SYNC:ENABle
value: EnableStruct = driver.status.questionable.sync.enable.get()
```

No command help available

```
return
    structure: for return value, see the help for EnableStruct structure arguments.
```

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTionable:SYNC:ENABle
driver.status.questionable.sync.enable.set(bit_definition = 1, channel_name = '1
↪')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.17.3 Event

SCPI Commands

```
STATUS:QUESTIONABLE:SYNC:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONABLE:SYNC[:EVENT]
value: str = driver.status.questionable.sync.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.17.4 Ntransition

SCPI Commands

```
STATUS:QUESTIONABLE:SYNC:NTRANSITION
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:SYNC:NTRANSITION
value: NtransitionStruct = driver.status.questionable.sync.ntransition.get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:SYNC:NTRansition
driver.status.questionable.sync.ntransition.set(bit_definition = 1, channel_
↳name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.17.5 Ptransition

SCPI Commands

```
STATus:QUESTionable:SYNC:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATus:QUESTionable:SYNC:PTRansition
value: PtransitionStruct = driver.status.questionable.sync.ptransition.get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:SYNC:PTRansition
driver.status.questionable.sync.ptransition.set(bit_definition = 1, channel_
↳name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.18 Temperature

class TemperatureCls

Temperature commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.temperature.clone()
```

Subgroups

6.19.2.18.1 Condition

SCPI Commands

```
STATus:QUEStionable:TEMPerature:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:TEMPerature:CONDition
value: str = driver.status.questionable.temperature.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.18.2 Enable

SCPI Commands

```
STATus:QUEStionable:TEMPerature:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUESTionable:TEMPerature:ENABle
value: EnableStruct = driver.status.questionable.temperature.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUESTionable:TEMPerature:ENABle
driver.status.questionable.temperature.enable.set(summary_bit = 1, channel_name_
↳= '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.18.3 Event

SCPI Commands

```
STATus:QUESTionable:TEMPerature:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUESTionable:TEMPerature[:EVENT]
value: str = driver.status.questionable.temperature.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.18.4 Ntransition

SCPI Commands

```
STATus:QuesTionable:TEMPerature:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QuesTionable:TEMPerature:NTRansition
value: NtransitionStruct = driver.status.questionable.temperature.ntransition.
↳get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QuesTionable:TEMPerature:NTRansition
driver.status.questionable.temperature.ntransition.set(summary_bit = 1, channel_
↳name = '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.18.5 Ptransition

SCPI Commands

`STATus:QUEStionable:TEMPerature:PTRansition`

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- `Summary_Bit`: int: No parameter help available
- `Channel_Name`: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:TEMPerature:PTRansition
value: PtransitionStruct = driver.status.questionable.temperature.ptransition.
↳get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:TEMPerature:PTRansition
driver.status.questionable.temperature.ptransition.set(summary_bit = 1, channel_
↳name = '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.19 Time

class TimeCls

Time commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.time.clone()
```

Subgroups

6.19.2.19.1 Condition

SCPI Commands

```
STATus:QUEStionable:TIME:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:TIME:CONDition
value: str = driver.status.questionable.time.condition.get()
```

These commands read out the CONDition section of the status register. The commands do not delete the contents of the CONDition section.

return

channel_name: String containing the name of the channel. The parameter is optional.
If you omit it, the command works for the currently active channel.

6.19.2.19.2 Enable

SCPI Commands

```
STATus:QUEStionable:TIME:ENABLe
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Summary_Bit: int: No parameter help available
- Channel_Name: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → EnableStruct

```
# SCPI: STATus:QUEStionable:TIME:ENABle
value: EnableStruct = driver.status.questionable.time.enable.get()
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

return

structure: for return value, see the help for EnableStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:TIME:ENABle
driver.status.questionable.time.enable.set(summary_bit = 1, channel_name = '1')
```

These commands control the ENABLE part of a register. The ENABLE part allows true conditions in the EVENT part of the status register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the summary bit reported to the next higher level.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.19.3 Event

SCPI Commands

```
STATus:QUEStionable:TIME:EVENTt
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATus:QUEStionable:TIME[:EVENTt]
value: str = driver.status.questionable.time.event.get()
```

These commands read out the EVENT section of the status register. At the same time, the commands delete the contents of the EVENT section.

return

channel_name: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.19.4 Ntransition

SCPI Commands

```
STATus:QUEStionable:TIME:NTRansition
```

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → NtransitionStruct

```
# SCPI: STATus:QUEStionable:TIME:NTRansition
value: NtransitionStruct = driver.status.questionable.time.ntransition.get()
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:TIME:NTRansition
driver.status.questionable.time.ntransition.set(summary_bit = 1, channel_name =
↪ '1')
```

These commands control the Negative TRansition part of a register. Setting a bit causes a 1 to 0 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.19.5 Ptransition

SCPI Commands

```
STATus:QUEStionable:TIME:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- **Summary_Bit**: int: No parameter help available
- **Channel_Name**: str: String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

get() → PtransitionStruct

```
# SCPI: STATus:QUEStionable:TIME:PTRansition
value: PtransitionStruct = driver.status.questionable.time.ptransition.get()
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(summary_bit: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATus:QUEStionable:TIME:PTRansition
driver.status.questionable.time.ptransition.set(summary_bit = 1, channel_name =
↪ '1')
```

These commands control the Positive TRansition part of a register. Setting a bit causes a 0 to 1 transition in the corresponding bit of the associated register. The transition also writes a 1 into the associated bit of the corresponding EVENT register.

param summary_bit

No help available

param channel_name

String containing the name of the channel. The parameter is optional. If you omit it, the command works for the currently active channel.

6.19.2.20 Transducer

class TransducerCls

Transducer commands group definition. 5 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.questionable.transducer.clone()
```

Subgroups

6.19.2.20.1 Condition

SCPI Commands

```
STATUS:QUESTionable:TRANsducer:CONDition
```

class ConditionCls

Condition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTionable:TRANsducer:CONDition
value: str = driver.status.questionable.transducer.condition.get()
```

No command help available

```
return
    channel_name: No help available
```

6.19.2.20.2 Enable

SCPI Commands

```
STATUS:QUESTionable:TRANsducer:ENABLE
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → EnableStruct

```
# SCPI: STATUS:QUESTionable:TRANsducer:ENABLE
value: EnableStruct = driver.status.questionable.transducer.enable.get()
```

No command help available

```
return
    structure: for return value, see the help for EnableStruct structure arguments.
```

set(bit_definition: int, channel_name: Optional[str] = None) → None

```
# SCPI: STATUS:QUESTionable:TRANsducer:ENABLE
driver.status.questionable.transducer.enable.set(bit_definition = 1, channel_
↪name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.20.3 Event

SCPI Commands

STATUS:QUESTIONABLE:TRANSDUCER:EVENT

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: STATUS:QUESTIONABLE:TRANSDUCER[:EVENT]
value: str = driver.status.questionable.transducer.event.get()
```

No command help available

return

channel_name: No help available

6.19.2.20.4 Ntransition

SCPI Commands

STATUS:QUESTIONABLE:TRANSDUCER:NTRANSITION

class NtransitionCls

Ntransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class NtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → NtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:TRANSDUCER:NTRANSITION
value: NtransitionStruct = driver.status.questionable.transducer.ntransition.
    ↪get()
```

No command help available

return

structure: for return value, see the help for NtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATUS:QUESTIONABLE:TRANSducer:NTRansition
driver.status.questionable.transducer.ntransition.set(bit_definition = 1,
↪channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.2.20.5 Ptransition

SCPI Commands

```
STATUS:QUESTIONABLE:TRANSducer:PTRansition
```

class PtransitionCls

Ptransition commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PtransitionStruct

Response structure. Fields:

- Bit_Definition: int: No parameter help available
- Channel_Name: str: No parameter help available

get() → PtransitionStruct

```
# SCPI: STATUS:QUESTIONABLE:TRANSducer:PTRansition
value: PtransitionStruct = driver.status.questionable.transducer.ptransition.
↪get()
```

No command help available

return

structure: for return value, see the help for PtransitionStruct structure arguments.

set(*bit_definition: int, channel_name: Optional[str] = None*) → None

```
# SCPI: STATUS:QUESTIONABLE:TRANSducer:PTRansition
driver.status.questionable.transducer.ptransition.set(bit_definition = 1,
↪channel_name = '1')
```

No command help available

param bit_definition

No help available

param channel_name

No help available

6.19.3 Queue

class QueueCls

Queue commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.status.queue.clone()
```

Subgroups

6.19.3.1 Next

SCPI Commands

```
STATUS:QUEue:NEXT
```

class NextCls

Next commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: STATUS:QUEue[:NEXT]
driver.status.queue.next.set()
```

This command queries the most recent error queue entry and deletes it. Positive error numbers indicate device-specific errors, negative error numbers are error messages defined by SCPI. If the error queue is empty, the error number 0, 'No error', is returned. This command is identical to the SYS-Tem:ERRor[:NEXT]? command.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: STATUS:QUEue[:NEXT]
driver.status.queue.next.set_with_opc()
```

This command queries the most recent error queue entry and deletes it. Positive error numbers indicate device-specific errors, negative error numbers are error messages defined by SCPI. If the error queue is empty, the error number 0, 'No error', is returned. This command is identical to the SYS-Tem:ERRor[:NEXT]? command.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20 System

class SystemCls

System commands group definition. 86 total commands, 29 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.clone()
```

Subgroups

6.20.1 Clogging

SCPI Commands

SYSTem:CLOGging

class CloggingCls

Clogging commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:CLOGging
value: bool = driver.system.clogging.get()
```

This command turns logging of remote commands on and off.

return

state: ON | OFF | 1 | 0 ON | 1 Writes all remote commands that have been sent to a file. The destination is C:/R_S/INSTR/ScpiLogging/ ScpiLog.no.. where no. is a sequential number A new log file is started each time logging was stopped and is restarted. OFF | 0

set(state: bool) → None

```
# SCPI: SYSTem:CLOGging
driver.system.clogging.set(state = False)
```

This command turns logging of remote commands on and off.

param state

ON | OFF | 1 | 0 ON | 1 Writes all remote commands that have been sent to a file. The destination is C:/R_S/INSTR/ScpiLogging/ ScpiLog.no.. where no. is a sequential number A new log file is started each time logging was stopped and is restarted. OFF | 0

6.20.2 Communicate

class CommunicateCls

Communicate commands group definition. 30 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.clone()
```

Subgroups

6.20.2.1 Gpib

class GpibCls

Gpib commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.gpib.clone()
```

Subgroups

6.20.2.1.1 Rdevice

class RdeviceCls

Rdevice commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.gpib.rdevice.clone()
```

Subgroups

6.20.2.1.1.1 Generator<Generator>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.system.communicate.gpib.rdevice.generator.repcap_generator_get()
driver.system.communicate.gpib.rdevice.generator.repcap_generator_set(repcap.Generator.
↪Nr1)
```

class GeneratorCls

Generator commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Generator, default value after init: Generator.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.gpib.rdevice.generator.clone()
```

Subgroups**6.20.2.1.1.2 Address****SCPI Commands**

```
SYSTEM:COMMunicate:GPIB:RDEvice:GENerator<Generator>:ADDRess
```

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(generator=Generator.Default) → int

```
# SCPI: SYSTEM:COMMunicate:GPIB:RDEvice:GENerator<gen>:ADDRess
value: int = driver.system.communicate.gpib.rdevice.generator.address.
↳get(generator = repcap.Generator.Default)
```

No command help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

return

number: No help available

set(number: int, generator=Generator.Default) → None

```
# SCPI: SYSTEM:COMMunicate:GPIB:RDEvice:GENerator<gen>:ADDRess
driver.system.communicate.gpib.rdevice.generator.address.set(number = 1,
↳generator = repcap.Generator.Default)
```

No command help available

param number

No help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

6.20.2.1.2 Self

class SelfCls

Self commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.gpib.self.clone()
```

Subgroups

6.20.2.1.2.1 Address

SCPI Commands

```
SYSTem:COMMunicate:GPIB:SELF:ADDRess
```

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: SYSTem:COMMunicate:GPIB[:SELF]:ADDRess
value: float = driver.system.communicate.gpib.self.address.get()
```

This command sets the GPIB address of the R&S FSWP.

return

address: Range: 0 to 30

set(address: float) → None

```
# SCPI: SYSTem:COMMunicate:GPIB[:SELF]:ADDRess
driver.system.communicate.gpib.self.address.set(address = 1.0)
```

This command sets the GPIB address of the R&S FSWP.

param address

Range: 0 to 30

6.20.2.1.2.2 Rterminator

SCPI Commands

```
SYSTem:COMMunicate:GPIB:SELF:RTERminator
```

class RterminatorCls

Rterminator commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → GpibTerminator

```
# SCPI: SYSTem:COMMunicate:GPIB[:SELF]:RTERminator
value: enums.GpibTerminator = driver.system.communicate.gpib.self.rterminator.
↪get()
```

This command selects the GPIB receive terminator. Output of binary data from the instrument to the control computer does not require such a terminator change.

return

terminator: LFEOI | EOI LFEOI According to the standard, the terminator in ASCII is LF and/or EOI. EOI For binary data transfers (e.g. trace data) from the control computer to the instrument, the binary code used for LF might be included in the binary data block, and therefore should not be interpreted as a terminator in this particular case. This can be avoided by using only the receive terminator EOI.

set(terminator: GpibTerminator) → None

```
# SCPI: SYSTem:COMMunicate:GPIB[:SELF]:RTERminator
driver.system.communicate.gpib.self.rterminator.set(terminator = enums.
↪GpibTerminator.EOI)
```

This command selects the GPIB receive terminator. Output of binary data from the instrument to the control computer does not require such a terminator change.

param terminator

LFEOI | EOI LFEOI According to the standard, the terminator in ASCII is LF and/or EOI. EOI For binary data transfers (e.g. trace data) from the control computer to the instrument, the binary code used for LF might be included in the binary data block, and therefore should not be interpreted as a terminator in this particular case. This can be avoided by using only the receive terminator EOI.

6.20.2.2 Internal

class InternalCls

Internal commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.internal.clone()
```

Subgroups

6.20.2.2.1 Command

class CommandCls

Command commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.internal.command.clone()
```

Subgroups

6.20.2.2.1.1 Tables

SCPI Commands

```
SYSTem:COMMunicate:INTernal:COMManD:TABLEs
```

class TablesCls

Tables commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:INTernal[:COMManD]:TABLEs
value: str = driver.system.communicate.internal.command.tables.get()
```

No command help available

return

channel_name: No help available

set(channel_name: str) → None

```
# SCPI: SYSTem:COMMunicate:INTernal[:COMManD]:TABLEs
driver.system.communicate.internal.command.tables.set(channel_name = '1')
```

No command help available

param channel_name

No help available

6.20.2.2.2 Completed

class CompletedCls

Completed commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.internal.completed.clone()
```

Subgroups

6.20.2.2.1 Event

SCPI Commands

```
SYSTem:COMMunicate:INTernal:COMPLeted:EVENT
```

class EventCls

Event commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:INTernal[:COMPLeted]:EVENT
value: str = driver.system.communicate.internal.completed.event.get()
```

No command help available

```
return
    channel_name: No help available
```

set(channel_name: str) → None

```
# SCPI: SYSTem:COMMunicate:INTernal[:COMPLeted]:EVENT
driver.system.communicate.internal.completed.event.set(channel_name = '1')
```

No command help available

```
param channel_name
    No help available
```

6.20.2.3 Rdevice

class RdeviceCls

Rdevice commands group definition. 14 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.clone()
```

Subgroups

6.20.2.3.1 Generator<Generator>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.system.communicate.rdevice.generator.repcap_generator_get()
driver.system.communicate.rdevice.generator.repcap_generator_set(repcap.Generator.Nr1)
```

class GeneratorCls

Generator commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: Generator, default value after init: Generator.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.generator.clone()
```

Subgroups**6.20.2.3.1.1 Interface****SCPI Commands**

```
SYSTEM:COMMunicate:RDEVice:GENerator<Generator>:INTERface
```

class InterfaceCls

Interface commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(generator=*Generator.Default*) → GeneratorIntfType

```
# SCPI: SYSTEM:COMMunicate:RDEVice:GENerator<gen>:INTERface
value: enums.GeneratorIntfType = driver.system.communicate.rdevice.generator.
↳ interface.get(generator = repcap.Generator.Default)
```

No command help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

return

type_py: No help available

set(type_py: GeneratorIntfType, generator=*Generator.Default*) → None

```
# SCPI: SYSTEM:COMMunicate:RDEVice:GENerator<gen>:INTERface
driver.system.communicate.rdevice.generator.interface.set(type_py = enums.
↳ GeneratorIntfType.GPIB, generator = repcap.Generator.Default)
```

No command help available

param type_py

No help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

6.20.2.3.1.2 Link

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:GENerator<Generator>:LINK
```

class LinkCls

Link commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(generator=*Generator.Default*) → GeneratorLink

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator<gen>:LINK
value: enums.GeneratorLink = driver.system.communicate.rdevice.generator.link.
↳get(generator = repcap.Generator.Default)
```

No command help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

return

type_py: No help available

set(type_py: *GeneratorLink*, generator=*Generator.Default*) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator<gen>:LINK
driver.system.communicate.rdevice.generator.link.set(type_py = enums.
↳GeneratorLink.GPIB, generator = repcap.Generator.Default)
```

No command help available

param type_py

No help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

6.20.2.3.1.3 TypePy

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:GENerator<Generator>:TYPE
```

class TypePyCls

TypePy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(generator=*Generator.Default*) → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator<gen>:TYPE
value: str = driver.system.communicate.rdevice.generator.typePy.get(generator =
↳repcap.Generator.Default)
```

No command help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

return

name: No help available

set(name: str, generator=Generator.Default) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:GENerator<gen>:TYPE
driver.system.communicate.rdevice.generator.typePy.set(name = '1', generator = ↵
↵repcap.Generator.Default)
```

No command help available

param name

No help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

6.20.2.3.2 Oscilloscope

class OscilloscopeCls

Oscilloscope commands group definition. 8 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.oscilloscope.clone()
```

Subgroups

6.20.2.3.2.1 Alignment

class AlignmentCls

Alignment commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.oscilloscope.alignment.clone()
```

Subgroups

6.20.2.3.2.2 Date

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:ALIGnment:DATE
```

class DateCls

Date commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:ALIGnment:DATE
value: str = driver.system.communicate.rdevice.oscilloscope.alignment.date.get()
```

No command help available

```
return
    alignment_date: No help available
```

6.20.2.3.2.3 Step<Step>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.system.communicate.rdevice.oscilloscope.alignment.step.repcap_step_get()
driver.system.communicate.rdevice.oscilloscope.alignment.step.repcap_step_set(repcap.
↳ Step.Nr1)
```

class StepCls

Step commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Step, default value after init: Step.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.oscilloscope.alignment.step.clone()
```

Subgroups

6.20.2.3.2.4 State

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:ALIGnment:STEP<Step>:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*step*=*Step.Default*) → bool

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:ALIGnment:STEP<1|2>[:STATE]
value: bool = driver.system.communicate.rdevice.oscilloscope.alignment.step.
↪state.get(step = repcap.Step.Default)
```

No command help available

param step

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Step’)

return

alignment_step: No help available

6.20.2.3.2.5 Idn

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:IDN
```

class IdnCIs

Idn commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:IDN
value: str = driver.system.communicate.rdevice.oscilloscope.idn.get()
```

No command help available

return

idn: No help available

6.20.2.3.2.6 LedState

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:LEDState
```

class LedStateCIs

LedState commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:LEDState
value: str = driver.system.communicate.rdevice.oscilloscope.ledState.get()
```

No command help available

return

led_state: No help available

6.20.2.3.2.7 State

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:STATe
value: bool = driver.system.communicate.rdevice.oscilloscope.state.get()
```

No command help available

return
state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:STATe
driver.system.communicate.rdevice.oscilloscope.state.set(state = False)
```

No command help available

param state
No help available

6.20.2.3.2.8 TcpiP

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:OSCilloscope:TCPIp
```

class TcpiPcls

TcpiP commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:TCPIp
value: str = driver.system.communicate.rdevice.oscilloscope.tcpiP.get()
```

No command help available

return
tcpiP: No help available

set(tcpiP: str) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:TCPIp
driver.system.communicate.rdevice.oscilloscope.tcpiP.set(tcpiP = '1')
```

No command help available

param tcpip
No help available

6.20.2.3.2.9 Vdevice

SCPI Commands

SYSTem:COMMunicate:RDEvice:OSCilloscope:VDEvice

class VdeviceCls

Vdevice commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:VDEvice
value: bool = driver.system.communicate.rdevice.oscilloscope.vdevice.get()
```

No command help available

return
valid_device: No help available

set(valid_device: bool) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:VDEvice
driver.system.communicate.rdevice.oscilloscope.vdevice.set(valid_device = False)
```

No command help available

param valid_device
No help available

6.20.2.3.2.10 Vfirmware

SCPI Commands

SYSTem:COMMunicate:RDEvice:OSCilloscope:VFIRmware

class VfirmwareCls

Vfirmware commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:VFIRmware
value: bool = driver.system.communicate.rdevice.oscilloscope.vfirmware.get()
```

No command help available

return
valid_device: No help available

set(valid_device: bool) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:OSCilloscope:VFIRmware
driver.system.communicate.rdevice.oscilloscope.vfirmware.set(valid_device =
↪False)
```

No command help available

param valid_device

No help available

6.20.2.3.3 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.system.communicate.rdevice.pmeter.repcap_powerMeter_get()
driver.system.communicate.rdevice.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.pmeter.clone()
```

Subgroups

6.20.2.3.3.1 Configure

class ConfigureCls

Configure commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.pmeter.configure.clone()
```

Subgroups

6.20.2.3.3.2 Auto

class AutoCls

Auto commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rdevice.pmeter.configure.auto.clone()
```

Subgroups

6.20.2.3.3.3 State

SCPI Commands

```
SYSTem:COMMunicate:RDEvice:PMETer<PowerMeter>:CONFigure:AUTO:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(powerMeter=PowerMeter.Default) → bool

```
# SCPI: SYSTem:COMMunicate:RDEvice:PMETer<p>:CONFigure:AUTO[:STATe]
value: bool = driver.system.communicate.rdevice.pmeter.configure.auto.state.
↳ get(powerMeter = repcap.PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

state: No help available

set(state: bool, powerMeter=PowerMeter.Default) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:PMETer<p>:CONFigure:AUTO[:STATe]
driver.system.communicate.rdevice.pmeter.configure.auto.state.set(state = False,
↳ powerMeter = repcap.PowerMeter.Default)
```

No command help available

param state

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.20.2.3.3.4 Count

SCPI Commands

```
SYSTEM:COMMunicate:RDEvice:PMETer<PowerMeter>:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → float

```
# SCPI: SYSTEM:COMMunicate:RDEvice:PMETer<p>:COUNT
value: float = driver.system.communicate.rdevice.pmeter.count.get(powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

number_sensors: No help available

6.20.2.3.3.5 Define

SCPI Commands

```
SYSTEM:COMMunicate:RDEvice:PMETer<PowerMeter>:DEFine
```

class DefineCls

Define commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class DefineStruct

Response structure. Fields:

- Placeholder: str: No parameter help available
- Type_Py: str: No parameter help available
- Interface: str: No parameter help available
- Serial_No: str: No parameter help available

get(*powerMeter=PowerMeter.Default*) → DefineStruct

```
# SCPI: SYSTEM:COMMunicate:RDEvice:PMETer<p>:DEFine
value: DefineStruct = driver.system.communicate.rdevice.pmeter.define.
↳get(powerMeter = repcap.PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

structure: for return value, see the help for DefineStruct structure arguments.

set(placeholder: str, type_py: str, interface: str, serial_no: str, powerMeter=PowerMeter.Default) → None

```
# SCPI: SYSTem:COMMunicate:RDEvice:PMETer<p>:DEFine
driver.system.communicate.rdevice.pmeter.define.set(placeholder = '1', type_py_
↳= '1', interface = '1', serial_no = '1', powerMeter = repcap.PowerMeter.
↳Default)
```

No command help available

param placeholder

No help available

param type_py

No help available

param interface

No help available

param serial_no

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.20.2.4 Rest

class RestCls

Rest commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.rest.clone()
```

Subgroups

6.20.2.4.1 Enable

SCPI Commands

```
SYSTem:COMMunicate:REST:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:COMMunicate:REST:ENABle
value: bool = driver.system.communicate.rest.enable.get()
```

Turns communication via the REST API on and off.

return

state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:COMMunicate:REST:ENABle
driver.system.communicate.rest.enable.set(state = False)
```

Turns communication via the REST API on and off.

param state

ON | OFF | 0 | 1

6.20.2.5 Snmp

class SnmpCls

Snmp commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.snmp.clone()
```

Subgroups

6.20.2.5.1 Community

class CommunityCls

Community commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.snmp.community.clone()
```

Subgroups

6.20.2.5.1.1 Ro

SCPI Commands

```
SYSTem:COMMunicate:SNMP:COMMunity:RO
```

class RoCls

Ro commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(community: str) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:COMMunity:RO
driver.system.communicate.snmp.community.ro.set(community = '1')
```

Defines the SNMP community string for read-only access.

INTRO_CMD_HELP: Prerequisites for this command:

- Select an SNMP version that supports communities (method RsFswp.System.Communicate.Snmp.Version.set) .

param community

String containing the community name.

6.20.2.5.1.2 Rw

SCPI Commands

```
SYSTem:COMMunicate:SNMP:COMMunity:RW
```

class RwCls

Rw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(community: str) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:COMMunity:RW
driver.system.communicate.snmp.community.rw.set(community = '1')
```

Defines the SNMP community string for read-write access.

INTRO_CMD_HELP: Prerequisites for this command:

- Select an SNMP version that supports communities (method RsFswp.System.Communicate.Snmp.Version.set) .

param community

String containing the community name.

6.20.2.5.2 Contact

SCPI Commands

```
SYSTem:COMMunicate:SNMP:CONtact
```

class ContactCls

Contact commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:SNMP:CONtact
value: str = driver.system.communicate.snmp.contact.get()
```

Defines the SNMP contact information for the administrator. You can also set the contact information via SNMP if you do not set it via SCPI.

return
contact_info: No help available

set(contact_info: str) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:CONtact
driver.system.communicate.snmp.contact.set(contact_info = '1')
```

Defines the SNMP contact information for the administrator. You can also set the contact information via SNMP if you do not set it via SCPI.

param contact_info
String containing SNMP contact.

6.20.2.5.3 Location

SCPI Commands

```
SYSTem:COMMunicate:SNMP:LOCation
```

class LocationCls

Location commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:COMMunicate:SNMP:LOCation
value: str = driver.system.communicate.snmp.location.get()
```

Defines the SNMP location information for the administrator. You can also set the location information via SNMP if you do not set it via SCPI.

return
location: No help available

set(location: str) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:LOCation
driver.system.communicate.snmp.location.set(location = '1')
```

Defines the SNMP location information for the administrator. You can also set the location information via SNMP if you do not set it via SCPI.

param location
String containing SNMP location.

6.20.2.5.4 Usm

class UsmCls

Usm commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.snmp.usm.clone()
```

Subgroups

6.20.2.5.4.1 User

SCPI Commands

```
SYSTem:COMMunicate:SNMP:USM:USER
SYSTem:COMMunicate:SNMP:USM:USER:DELeTe
SYSTem:COMMunicate:SNMP:USM:USER:DELeTe:ALL
```

class UserCls

User commands group definition. 4 total commands, 1 Subgroups, 3 group commands

delete(name: str) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:USM:USER:DELeTe
driver.system.communicate.snmp.usm.user.delete(name = '1')
```

Deletes a specific SNMP user profile.

param name

String containing name of SNMP user profile to be deleted.

delete_all() → None

```
# SCPI: SYSTem:COMMunicate:SNMP:USM:USER:DELeTe:ALL
driver.system.communicate.snmp.usm.user.delete_all()
```

Deletes all SNMP user profiles.

delete_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:USM:USER:DELeTe:ALL
driver.system.communicate.snmp.usm.user.delete_all_with_opc()
```

Deletes all SNMP user profiles.

Same as delete_all, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

set(name: str, access: AccessType, level: UserLevel, auth_pwd: Optional[str] = None, priv_pwd: Optional[str] = None) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:USM:USER
driver.system.communicate.snmp.usm.user.set(name = '1', access = enums.
↳ AccessType.R0, level = enums.UserLevel.AUTH, auth_pwd = '1', priv_pwd = '1')
```

Defines an SNMP user profile.

INTRO_CMD_HELP: Prerequisites for this command:

- Select SNMPv3 (method RsFswp.System.Communicate.Snmp.Version.set) .

param name

String containing name of the user.

param access

RO | RW Defines the access right a user can have.

param level

NOAuth | AUTH | PRIVacy Defines the security level.

param auth_pwd

String containing the authentication password.

param priv_pwd

String containing the privacy password.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.snmp.usm.user.clone()
```

Subgroups

6.20.2.5.4.2 All

SCPI Commands

```
SYSTem:COMMunicate:SNMP:USM:USER:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Count: float: Total number of registered SNMP users.
- Name: str: List of all user names as a comma-separated list.

get() → GetStruct

```
# SCPI: SYSTem:COMMunicate:SNMP:USM:USER:ALL
value: GetStruct = driver.system.communicate.snmp.usm.user.all.get()
```

Queries the number of users and a list of all SNMP users for SNMPv3.

INTRO_CMD_HELP: Prerequisites for this command:

- Select SNMPv3 (method RsFswp.System.Communicate.Snmp.Version.set) .

return

structure: for return value, see the help for GetStruct structure arguments.

6.20.2.5.5 Version**SCPI Commands**

SYSTem:COMMunicate:SNMP:VERSion

class VersionCls

Version commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SnmpVersion

```
# SCPI: SYSTem:COMMunicate:SNMP:VERSion
value: enums.SnmpVersion = driver.system.communicate.snmp.version.get()
```

Selects the SNMP version.

return

snmp_version: OFF | V12 | V123 | V3 | DEFault OFF SNMP communication is off. V12 SNMP communication with SNMPv2 or lower. V123 SNMP communication with SNMPv2 and SNMPv3. V3 SNMP communication with SNMPv3.

set(snmpp_version: SnmpVersion) → None

```
# SCPI: SYSTem:COMMunicate:SNMP:VERSion
driver.system.communicate.snmp.version.set(snmpp_version = enums.SnmpVersion.
↳DEFault)
```

Selects the SNMP version.

param snmp_version

OFF | V12 | V123 | V3 | DEFault OFF SNMP communication is off. V12 SNMP communication with SNMPv2 or lower. V123 SNMP communication with SNMPv2 and SNMPv3. V3 SNMP communication with SNMPv3.

6.20.2.6 Tcpiip**class TcpiipCls**

Tcpiip commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.tcpip.clone()
```

Subgroups

6.20.2.6.1 Rdevice

class RdeviceCls

Rdevice commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.tcpip.rdevice.clone()
```

Subgroups

6.20.2.6.1.1 Generator<Generator>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.system.communicate.tcpip.rdevice.generator.repcap_generator_get()
driver.system.communicate.tcpip.rdevice.generator.repcap_generator_set(repcap.Generator.
↳Nr1)
```

class GeneratorCls

Generator commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Generator, default value after init: Generator.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.communicate.tcpip.rdevice.generator.clone()
```

Subgroups

6.20.2.6.1.2 Address

SCPI Commands

```
SYSTem:COMMunicate:TCPIp:RDEvice:GENerator<Generator>:ADDress
```

class AddressCls

Address commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(generator=*Generator.Default*) → str

```
# SCPI: SYSTem:COMMunicate:TCPIp:RDEvice:GENerator<gen>:ADDRess
value: str = driver.system.communicate.tcpip.rdevice.generator.address.
↳get(generator = reprcap.Generator.Default)
```

No command help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

return

address: No help available

set(address: str, generator=*Generator.Default*) → None

```
# SCPI: SYSTem:COMMunicate:TCPIp:RDEvice:GENerator<gen>:ADDRess
driver.system.communicate.tcpip.rdevice.generator.address.set(address = '1',
↳generator = reprcap.Generator.Default)
```

No command help available

param address

No help available

param generator

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Generator')

6.20.3 Compatible

SCPI Commands

```
SYSTem:COMPatible
```

class CompatibleCls

Compatible commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CompatibilityMode

```
# SCPI: SYSTem:COMPatible
value: enums.CompatibilityMode = driver.system.compatible.get()
```

No command help available

return

mode: No help available

set(mode: *CompatibilityMode*) → None

```
# SCPI: SYSTem:COMPatible
driver.system.compatible.set(mode = enums.CompatibilityMode.ATT)
```

No command help available

param mode

No help available

6.20.4 DeviceFootprint

SCPI Commands

```
SYSTem:DFPRint
```

class DeviceFootprintCls

DeviceFootprint commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:DFPRint
value: str = driver.system.deviceFootprint.get()
```

No command help available

return

xxx: No help available

6.20.5 Display

class DisplayCls

Display commands group definition. 5 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.display.clone()
```

Subgroups

6.20.5.1 Fpanel

class FpanelCls

Fpanel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.display.fpanel.clone()
```

Subgroups

6.20.5.1.1 State

SCPI Commands

```
SYSTem:DISPlay:FPANel:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:DISPlay:FPANel[:STATe]
value: bool = driver.system.display.fpanel.state.get()
```

This command includes or excludes the front panel keys when working with the remote desktop.

return

state: ON | OFF | 0 | 1

set(state: bool) → None

```
# SCPI: SYSTem:DISPlay:FPANel[:STATe]
driver.system.display.fpanel.state.set(state = False)
```

This command includes or excludes the front panel keys when working with the remote desktop.

param state

ON | OFF | 0 | 1

6.20.5.2 Lock

SCPI Commands

```
SYSTem:DISPlay:LOCK
```

class LockCls

Lock commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:DISPlay:LOCK
value: bool = driver.system.display.lock.get()
```

Defines whether the ‘Display Update’ function remains available in remote operation or not.

return

state: ON | OFF | 0 | 1 OFF | 0 The function remains available. ON | 1 The function is not available and the display is not updated during remote operation.

set(state: bool) → None

```
# SCPI: SYSTem:DISPlay:LOCK
driver.system.display.lock.set(state = False)
```

Defines whether the 'Display Update' function remains available in remote operation or not.

param state

ON | OFF | 0 | 1 OFF | 0 The function remains available. ON | 1 The function is not available and the display is not updated during remote operation.

6.20.5.3 Message

class MessageCls

Message commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.display.message.clone()
```

Subgroups

6.20.5.3.1 State

SCPI Commands

```
SYSTem:DISPlay:MESSage:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:DISPlay:MESSage:STATe
value: bool = driver.system.display.message.state.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:DISPlay:MESSage:STATe
driver.system.display.message.state.set(state = False)
```

No command help available

param state
No help available

6.20.5.3.2 Text

SCPI Commands

SYSTem:DISPlay:MESSage:TEXT

class TextCls

Text commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:DISPlay:MESSage[:TEXT]
value: str = driver.system.display.message.text.get()
```

No command help available

return
message: No help available

set(message: str) → None

```
# SCPI: SYSTem:DISPlay:MESSage[:TEXT]
driver.system.display.message.text.set(message = '1')
```

No command help available

param message
No help available

6.20.5.4 Update

SCPI Commands

SYSTem:DISPlay:UPDate

class UpdateCls

Update commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:DISPlay:UPDate
value: bool = driver.system.display.update.get()
```

This command turns the display during remote operation on and off. If on, the R&S FSWP updates the diagrams, traces and display fields only. The best performance is obtained if the display is off during remote control operation.

return
state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SYSTem:DISPlay:UPDate
driver.system.display.update.set(state = False)
```

This command turns the display during remote operation on and off. If on, the R&S FSWP updates the diagrams, traces and display fields only. The best performance is obtained if the display is off during remote control operation.

param state
ON | OFF | 1 | 0

6.20.6 Error

SCPI Commands

```
SYSTem:ERRor:CLEar:ALL
```

class ErrorCls

Error commands group definition. 5 total commands, 4 Subgroups, 1 group commands

clear_all() → None

```
# SCPI: SYSTem:ERRor:CLEar:ALL
driver.system.error.clear_all()
```

This command deletes all contents of the ‘System Messages’ table.

clear_all_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:ERRor:CLEar:ALL
driver.system.error.clear_all_with_opc()
```

This command deletes all contents of the ‘System Messages’ table.

Same as clear_all, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.error.clone()
```

Subgroups

6.20.6.1 All

SCPI Commands

SYSTem:ERRor:ALL

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: SYSTem:ERRor:ALL
value: List[str] = driver.system.error.all.get()
```

No command help available

```
return
    result: No help available
```

6.20.6.2 Clear

class ClearCls

Clear commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.error.clear.clone()
```

Subgroups

6.20.6.2.1 Remote

SCPI Commands

SYSTem:ERRor:CLear:REMOte

class RemoteCls

Remote commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:ERRor:CLear:REMOte
driver.system.error.clear.remote.set()
```

This command deletes all contents of the ‘Remote Errors’ table. Note: The remote error list is automatically cleared when the R&S FSWP is shut down.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:ERRor:CLEar:REMoTe
driver.system.error.clear.remote.set_with_opc()
```

This command deletes all contents of the ‘Remote Errors’ table. Note: The remote error list is automatically cleared when the R&S FS WP is shut down.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.6.3 Display

SCPI Commands

```
SYSTem:ERRor:DISPlay
```

class DisplayCls

Display commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:ERRor:DISPlay
value: bool = driver.system.error.display.get()
```

This command switches the error display during remote operation on and off. If activated, the R&S FS WP displays a message box at the bottom of the screen that contains the most recent type of error and the command that caused the error.

return

state: ON | OFF | 1 | 0

set(state: bool) → None

```
# SCPI: SYSTem:ERRor:DISPlay
driver.system.error.display.set(state = False)
```

This command switches the error display during remote operation on and off. If activated, the R&S FS WP displays a message box at the bottom of the screen that contains the most recent type of error and the command that caused the error.

param state

ON | OFF | 1 | 0

6.20.6.4 ListPy

SCPI Commands

```
SYSTem:ERRor:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- System_Messages: str: String containing all messages in the ‘System Messages’ table.
- Remote_Errors: str: Error_no| Description| Command| Date| Time Comma-separated list of errors from the ‘Remote Errors’ table, where: Error_no: device-specific error code Description: brief description of the error Command: remote command causing the error Date|Time: date and time the error occurred

get(mess_type: Optional[MessageType] = None) → GetStruct

```
# SCPI: SYSTem:ERRor:LIST
value: GetStruct = driver.system.error.listPy.get(mess_type = enums.MessageType.
↳ REMote)
```

This command queries the error messages that occur during R&S FSWP operation.

param mess_type

SMSG | REMote SMSG (default) Queries the system messages which occurred during manual operation. REMote Queries the error messages that occurred during remote operation. Note: The remote error list is automatically cleared when the R&S FSWP is shut down.

return

structure: for return value, see the help for GetStruct structure arguments.

6.20.7 Firmware

class FirmwareCls

Firmware commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.firmware.clone()
```

Subgroups

6.20.7.1 Update

SCPI Commands

```
SYSTem:FIRMware:UPDate
```

class UpdateCls

Update commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:FIRMware:UPDate
value: str = driver.system.firmware.update.get()
```

This command starts a firmware update using the *.msi files in the selected directory. The default path is D:/FW_UPDATE. The path is changed via the method RsFswp.MassMemory.Comment.set command. To store the update files the MMEMemory:DATA command is used. Only user accounts with administrator rights can perform a firmware update.

return
directory: No help available

set(directory: str) → None

```
# SCPI: SYSTem:FIRMware:UPDate
driver.system.firmware.update.set(directory = '1')
```

This command starts a firmware update using the *.msi files in the selected directory. The default path is D:/FW_UPDATE. The path is changed via the method RsFswp.MassMemory.Comment.set command. To store the update files the MMEMemory:DATA command is used. Only user accounts with administrator rights can perform a firmware update.

param directory
No help available

6.20.8 FormatPy

class FormatPyCls

FormatPy commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.formatPy.clone()
```

Subgroups

6.20.8.1 Ident

SCPI Commands

SYSTem:FORMat:IDENt

class IdentCls

Ident commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → IdnFormat

```
# SCPI: SYSTem:FORMat:IDENt
value: enums.IdnFormat = driver.system.formatPy.ident.get()
```

This command selects the response format to the **IDN?* query.

return

idn_format: LEGacy Format is compatible to R&S FSP/FSU/FSQ/FSG family. NEW
| FSL R&S FSWP format Format is also compatible to the R&S FSL and R&S FSV
family

set(idn_format: IdnFormat) → None

```
# SCPI: SYSTem:FORMat:IDENt
driver.system.formatPy.ident.set(idn_format = enums.IdnFormat.FSL)
```

This command selects the response format to the **IDN?* query.

param idn_format

LEGacy Format is compatible to R&S FSP/FSU/FSQ/FSG family. NEW | FSL R&S
FSWP format Format is also compatible to the R&S FSL and R&S FSV family

6.20.9 Help

class HelpCls

Help commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.help.clone()
```

Subgroups

6.20.9.1 Headers

SCPI Commands

SYSTem:HELP:HEADers

class HeadersCls

Headers commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: SYSTem:HELP:HEADers
value: bytes = driver.system.help.headers.get()
```

No command help available

```
return
    header: No help available
```

6.20.9.2 Syntax

SCPI Commands

SYSTem:HELP:SYNTAX

class SyntaxCls

Syntax commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(command: str) → bytes

```
# SCPI: SYSTem:HELP:SYNTAX
value: bytes = driver.system.help.syntax.get(command = '1')
```

No command help available

```
param command
    No help available

return
    syntax: No help available
```

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.help.syntax.clone()
```

Subgroups

6.20.9.2.1 All

SCPI Commands

SYSTem:HELP:SYNTAX:ALL

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

```
# SCPI: SYSTem:HELP:SYNTAX:ALL
value: bytes = driver.system.help.syntax.all.get()
```

No command help available

return
syntax: No help available

6.20.10 Identify

class IdentifyCls

Identify commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.identify.clone()
```

Subgroups

6.20.10.1 Factory

SCPI Commands

SYSTem:IDENtify:FACTory

class FactoryCls

Factory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:IDENtify:FACTory
driver.system.identify.factory.set()
```

This command resets the query to ***IDN?** to its default value.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:IDENtify:FACTory
driver.system.identify.factory.set_with_opc()
```

This command resets the query to *IDN? to its default value.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.10.2 String

SCPI Commands

```
SYSTem:IDENtify:STRing
```

class StringCls

String commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:IDENtify[:STRing]
value: str = driver.system.identify.string.get()
```

This command defines the response to *IDN?.

return

string: String containing the description of the instrument.

set(string: str) → None

```
# SCPI: SYSTem:IDENtify[:STRing]
driver.system.identify.string.set(string = '1')
```

This command defines the response to *IDN?.

param string

String containing the description of the instrument.

6.20.11 IfGain

class IfGainCls

IfGain commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.ifGain.clone()
```

Subgroups

6.20.11.1 Mode

SCPI Commands

```
SYSTem:IFGain:MODE
```

class ModeCls

Mode commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → IfGainMode

```
# SCPI: SYSTem:IFGain:MODE
value: enums.IfGainMode = driver.system.ifGain.mode.get()
```

No command help available

return
mode: No help available

set(mode: IfGainMode) → None

```
# SCPI: SYSTem:IFGain:MODE
driver.system.ifGain.mode.set(mode = enums.IfGainMode.NORMAL)
```

No command help available

param mode
No help available

6.20.12 Language

SCPI Commands

```
SYSTem:LANGuage
```

class LanguageCls

Language commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:LANGuage
value: str = driver.system.language.get()
```

This command selects the system language.

return

language: String containing the name of the language. 'SCPI' SCPI language. 'PSA' PSA emulation. For a list of supported commands, see 'Reference: command set of emulated PSA models'.

set(language: str) → None

```
# SCPI: SYSTem:LANGuage
driver.system.language.set(language = '1')
```

This command selects the system language.

param language

String containing the name of the language. 'SCPI' SCPI language. 'PSA' PSA emulation. For a list of supported commands, see 'Reference: command set of emulated PSA models'.

6.20.13 Lxi

class LxiCls

Lxi commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.lxi.clone()
```

Subgroups

6.20.13.1 Info

SCPI Commands

```
SYSTem:LXI:INFO
```

class InfoCls

Info commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:LXI:INFO
value: str = driver.system.lxi.info.get()
```

No command help available

return

lxi_info: No help available

6.20.13.2 LanReset

SCPI Commands

SYSTem:LXI:LANReset

class LanResetCls

LanReset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:LXI:LANReset
driver.system.lxi.lanReset.set()
```

This command resets the LAN configuration, as well as the ‘LAN’ password and instrument description.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:LXI:LANReset
driver.system.lxi.lanReset.set_with_opc()
```

This command resets the LAN configuration, as well as the ‘LAN’ password and instrument description.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.13.3 Mdescription

SCPI Commands

SYSTem:LXI:MDEscription

class MdescriptionCls

Mdescription commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:LXI:MDEscription
value: str = driver.system.lxi.mdescription.get()
```

This command defines the ‘LAN’ instrument description.

return

description: String containing the instrument description.

set(description: str) → None

```
# SCPI: SYSTem:LXI:MDEscription
driver.system.lxi.mdescription.set(description = '1')
```

This command defines the ‘LAN’ instrument description.

param description

String containing the instrument description.

6.20.13.4 Password

SCPI Commands

```
SYSTem:LXI:PASSword
```

class PasswordCls

Password commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:LXI:PASSword
value: str = driver.system.lxi.password.get()
```

This command defines the 'LAN' password.

return

password: String containing the password.

set(password: str) → None

```
# SCPI: SYSTem:LXI:PASSword
driver.system.lxi.password.set(password = '1')
```

This command defines the 'LAN' password.

param password

String containing the password.

6.20.14 Option

class OptionCls

Option commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.option.clone()
```

Subgroups

6.20.14.1 License

class LicenseCls

License commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.option.license.clone()
```

Subgroups

6.20.14.1.1 ListPy

SCPI Commands

```
SYSTem:OPTion:LICense:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Option: str: No parameter help available
- State: enums.OptionState: No parameter help available
- Days: float: No parameter help available

get() → GetStruct

```
# SCPI: SYSTem:OPTion:LICense[:LIST]
value: GetStruct = driver.system.option.license.listPy.get()
```

No command help available

return

structure: for return value, see the help for GetStruct structure arguments.

6.20.14.2 Trial

class TrialCls

Trial commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.option.trial.clone()
```

Subgroups

6.20.14.2.1 ListPy

SCPI Commands

```
SYSTem:OPTion:TRIAal:LIST
```

class ListPyCls

ListPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[str]

```
# SCPI: SYSTem:OPTion:TRIAal:LIST
value: List[str] = driver.system.option.trial.listPy.get()
```

No command help available

```
return
    result: No help available
```

6.20.14.2.2 State

SCPI Commands

```
SYSTem:OPTion:TRIAal:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:OPTion:TRIAal[:STATe]
value: bool = driver.system.option.trial.state.get()
```

No command help available

```
return
    state: No help available
```

set(state: bool) → None

```
# SCPI: SYSTem:OPTion:TRIAal[:STATe]
driver.system.option.trial.state.set(state = False)
```

No command help available

```
param state
    No help available
```

6.20.15 Osystem

SCPI Commands

SYSTEM:OSYstem

class OsystemCls

Osystem commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:OSYstem
value: str = driver.system.osystem.get()
```

No command help available

```
return
    operating_system: No help available
```

6.20.16 Plugin

class PluginCls

Plugin commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.plugin.clone()
```

Subgroups

6.20.16.1 AppStarter

SCPI Commands

SYSTEM:PLUGin:APPStarter:DElete

class AppStarterCls

AppStarter commands group definition. 7 total commands, 6 Subgroups, 1 group commands

delete(application_group: str, display_name: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:DElete
driver.system.plugin.appStarter.delete(application_group = '1', display_name =
↪ '1')
```

No command help available

```
param application_group
    No help available
```

param display_name
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.plugin.appStarter.clone()
```

Subgroups

6.20.16.1.1 Directory

SCPI Commands

```
SYSTem:PLUGin:APPStarter:DIRectory
```

class DirectoryCls

Directory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:PLUGin:APPStarter:DIRectory
value: str = driver.system.plugin.appStarter.directory.get()
```

No command help available

return
working_dir: No help available

set(working_dir: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:DIRectory
driver.system.plugin.appStarter.directory.set(working_dir = '1')
```

No command help available

param working_dir
No help available

6.20.16.1.2 Execute

SCPI Commands

```
SYSTem:PLUGin:APPStarter:EXECute
```

class ExecuteCls

Execute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(application_group: str, display_name: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:EXECute
driver.system.plugin.appStarter.execute.set(application_group = '1', display_
↳ name = '1')
```

No command help available

param application_group

No help available

param display_name

No help available

6.20.16.1.3 Icon

SCPI Commands

```
SYSTem:PLUGin:APPStarter:ICON
```

class IconCls

Icon commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class IconStruct

Response structure. Fields:

- Icon_Path: str: No parameter help available
- Icon_Index: str: No parameter help available

get() → IconStruct

```
# SCPI: SYSTem:PLUGin:APPStarter:ICON
value: IconStruct = driver.system.plugin.appStarter.icon.get()
```

No command help available

return

structure: for return value, see the help for IconStruct structure arguments.

set(icon_path: str, icon_index: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:ICON
driver.system.plugin.appStarter.icon.set(icon_path = '1', icon_index = '1')
```

No command help available

param icon_path

No help available

param icon_index

No help available

6.20.16.1.4 Name

SCPI Commands

```
SYSTem:PLUGin:APPStarter:NAME
```

class NameCls

Name commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:PLUGin:APPStarter:NAME
value: str = driver.system.plugin.appStarter.name.get()
```

No command help available

```
return
    display_name: No help available
```

set(display_name: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:NAME
driver.system.plugin.appStarter.name.set(display_name = '1')
```

No command help available

```
param display_name
    No help available
```

6.20.16.1.5 Params

SCPI Commands

```
SYSTem:PLUGin:APPStarter:PARams
```

class ParamsCls

Params commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:PLUGin:APPStarter:PARams
value: str = driver.system.plugin.appStarter.params.get()
```

No command help available

```
return
    params: No help available
```

set(params: str) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:PARams
driver.system.plugin.appStarter.params.set(params = '1')
```

No command help available

param params
No help available

6.20.16.1.6 Select

SCPI Commands

SYSTem:PLUGin:APPStarter:SElect

class SelectCls

Select commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*application_group: str, display_name: str*) → None

```
# SCPI: SYSTem:PLUGin:APPStarter:SElect
driver.system.plugin.appStarter.select.set(application_group = '1', display_
↪name = '1')
```

No command help available

param application_group
No help available

param display_name
No help available

6.20.17 Preamp

SCPI Commands

SYSTem:PREamp

class PreampCls

Preamp commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PreampOption

```
# SCPI: SYSTem:PREamp
value: enums.PreampOption = driver.system.preamp.get()
```

No command help available

return
option: No help available

set(*option: PreampOption*) → None

```
# SCPI: SYSTem:PREamp
driver.system.preamp.set(option = enums.PreampOption.B23)
```

No command help available

param option
No help available

6.20.18 Preset

class PresetCls

Preset commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.preset.clone()
```

Subgroups

6.20.18.1 Channel

class ChannelCls

Channel commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.preset.channel.clone()
```

Subgroups

6.20.18.1.1 Exec

SCPI Commands

```
SYSTem:PRESet:CHANnel:EXEC
```

class ExecCls

Exec commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:PRESet:CHANnel[:EXEC]
driver.system.preset.channel.exec.set()
```

This command restores the default instrument settings in the current channel. Use INST:SEL to select the channel.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:PRESet:CHANnel[:EXEC]
driver.system.preset.channel.exec.set_with_opc()
```

This command restores the default instrument settings in the current channel. Use INST:SEL to select the channel.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.18.2 Compatible

SCPI Commands

SYSTem:PRESet:COMPAtible

class CompatibleCls

Compatible commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → PresetCompatible

```
# SCPI: SYSTem:PRESet:COMPAtible
value: enums.PresetCompatible = driver.system.preset.compatible.get()
```

This command defines the operating mode that is activated when you switch on the R&S FSWP or press the [PRESET] key.

return

arg_0: No help available

set(arg_0: PresetCompatible) → None

```
# SCPI: SYSTem:PRESet:COMPAtible
driver.system.preset.compatible.set(arg_0 = enums.PresetCompatible.MRECeiver)
```

This command defines the operating mode that is activated when you switch on the R&S FSWP or press the [PRESET] key.

param arg_0

SANalyzer Defines Signal and Spectrum Analyzer operating mode as the presetting.
OFF Selects the phase noise application as the default application (default value) .

6.20.18.3 Exec

SCPI Commands

SYSTem:PRESet:EXEC

class ExecCls

Exec commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:PRESet[:EXEC]
driver.system.preset.exec.set()
```

This command presets the R&S FSWP. It is identical to **RST*.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:PRESet[:EXEC]
driver.system.preset.exec.set_with_opc()
```

This command presets the R&S FSWP. It is identical to **RST*.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.18.4 InputPy

SCPI Commands

```
SYSTem:PRESet:INPut
```

class InputPyCls

InputPy commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → InputSelect

```
# SCPI: SYSTem:PRESet:INPut
value: enums.InputSelect = driver.system.preset.inputPy.get()
```

No command help available

return

default_input: No help available

set(default_input: InputSelect) → None

```
# SCPI: SYSTem:PRESet:INPut
driver.system.preset.inputPy.set(default_input = enums.InputSelect.INPut1)
```

No command help available

param default_input

No help available

6.20.19 Psa

class PsaCls

Psa commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.psa.clone()
```

Subgroups

6.20.19.1 Wideband

SCPI Commands

```
SYSTem:PSA:WIDeband
```

class WidebandCls

Wideband commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:PSA:WIDeband
value: bool = driver.system.psa.wideband.get()
```

This command defines which option is returned when the **OPT?* query is executed, depending on the state of the wideband option. It is only available for PSA89600 emulation.

return

state: ON | OFF | HIGH OFF The option is indicated as 'B7J' ON The 40 MHz wideband is used. The option is indicated as 'B7J, 140'. HIGH The 80 MHz wideband is used. The option is indicated as 'B7J, 122'.

set(state: bool) → None

```
# SCPI: SYSTem:PSA:WIDeband
driver.system.psa.wideband.set(state = False)
```

This command defines which option is returned when the **OPT?* query is executed, depending on the state of the wideband option. It is only available for PSA89600 emulation.

param state

ON | OFF | HIGH OFF The option is indicated as 'B7J' ON The 40 MHz wideband is used. The option is indicated as 'B7J, 140'. HIGH The 80 MHz wideband is used. The option is indicated as 'B7J, 122'.

6.20.20 Reboot

SCPI Commands

```
SYSTem:REBoot
```

class RebootCls

Reboot commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:REBoot
driver.system.reboot.set()
```

This command reboots the instrument, including the operating system.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:REBoot
driver.system.reboot.set_with_opc()
```

This command reboots the instrument, including the operating system.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.21 Revision

class RevisionCls

Revision commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.revision.clone()
```

Subgroups

6.20.21.1 Factory

SCPI Commands

```
SYSTem:REVision:FACTory
```

class FactoryCls

Factory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:REVision:FACTory
driver.system.revision.factory.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:REVision:FACTory
driver.system.revision.factory.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.21.2 String

SCPI Commands

SYSTem:REVision:STRing

class StringCls

String commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → str

```
# SCPI: SYSTem:REVision[:STRing]
value: str = driver.system.revision.string.get()
```

No command help available

return

name: No help available

set(name: str) → None

```
# SCPI: SYSTem:REVision[:STRing]
driver.system.revision.string.set(name = '1')
```

No command help available

param name

No help available

6.20.22 Rsweep

SCPI Commands

SYSTem:RSWeep

class RsweepCls

Rsweep commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:RSWeep
value: bool = driver.system.rsweep.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:RSweep
driver.system.rsweep.set(state = False)
```

No command help available

param state

No help available

6.20.23 Security

class SecurityCls

Security commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.security.clone()
```

Subgroups

6.20.23.1 State

SCPI Commands

```
SYSTem:SECurity:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:SECurity[:STATe]
value: bool = driver.system.security.state.get()
```

Activates or queries secure user mode. Note: Before you activate secure user mode, store any instrument settings that are required beyond the current session, such as predefined instrument settings, transducer files, or self-alignment data. Note: Initially after installation of the R&S FSWP-K33 option, secure user mode must be enabled manually once before remote control is possible. This is necessary to prompt for a change of passwords. For details on the secure user mode see ‘Protecting data using the secure user mode’.

return

state: ON | OFF | 0 | 1 ON | 1 The R&S FSWP automatically reboots and starts in secure user mode. In secure user mode, no data is written to the instrument’s internal solid-state drive. Data that the R&S FSWP normally stores on the solid-state drive is redirected to SDRAM. OFF | 0 The R&S FSWP is set to normal instrument mode. Data is stored to the internal solid-state drive. Note: this parameter is for query only. Secure user mode cannot be deactivated via remote operation.

set(state: bool) → None

```
# SCPI: SYSTem:SECurity[:STATE]
driver.system.security.state.set(state = False)
```

Activates or queries secure user mode. Note: Before you activate secure user mode, store any instrument settings that are required beyond the current session, such as predefined instrument settings, transducer files, or self-alignment data. Note: Initially after installation of the R&S FSWP-K33 option, secure user mode must be enabled manually once before remote control is possible. This is necessary to prompt for a change of passwords. For details on the secure user mode see ‘Protecting data using the secure user mode’.

param state

ON | OFF | 0 | 1 ON | 1 The R&S FSWP automatically reboots and starts in secure user mode. In secure user mode, no data is written to the instrument’s internal solid-state drive. Data that the R&S FSWP normally stores on the solid-state drive is redirected to SDRAM. OFF | 0 The R&S FSWP is set to normal instrument mode. Data is stored to the internal solid-state drive. Note: this parameter is for query only. Secure user mode cannot be deactivated via remote operation.

6.20.24 Sequencer

SCPI Commands

SYSTem:SEQuencer

class SequencerCls

Sequencer commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:SEQuencer
value: bool = driver.system.sequencer.get()
```

This command turns the Sequencer on and off. The Sequencer must be active before any other Sequencer commands (INIT:SEQ...) are executed, otherwise an error occurs. For details on the Sequencer see ‘The Sequencer Concept’. A detailed programming example is provided in ‘Programming Example: Performing a Sequence of Measurements’.

return

state: ON | OFF | 0 | 1 ON | 1 The Sequencer is activated and a sequential measurement is started immediately. OFF | 0 The Sequencer is deactivated. Any running sequential measurements are stopped. Further Sequencer commands (INIT:SEQ...) are not available.

set(state: bool) → None

```
# SCPI: SYSTem:SEQuencer
driver.system.sequencer.set(state = False)
```

This command turns the Sequencer on and off. The Sequencer must be active before any other Sequencer commands (INIT:SEQ...) are executed, otherwise an error occurs. For details on the Sequencer see ‘The Sequencer Concept’. A detailed programming example is provided in ‘Programming Example: Performing a Sequence of Measurements’.

param state

ON | OFF | 0 | 1 ON | 1 The Sequencer is activated and a sequential measurement is started immediately. OFF | 0 The Sequencer is deactivated. Any running sequential measurements are stopped. Further Sequencer commands (INIT:SEQ...) are not available.

6.20.25 Set

SCPI Commands

SYSTem:SET

class SetCls

Set commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bytes

<pre># SCPI: SYSTem:SET value: bytes = driver.system.set.get()</pre>
--

No command help available

return

block_data: No help available

set(block_data: bytes) → None

<pre># SCPI: SYSTem:SET driver.system.set.set(block_data = b'ABCDEFGH')</pre>

No command help available

param block_data

No help available

6.20.26 ShImmediate

SCPI Commands

SYSTem:SHIMmediate

class ShImmediateCls

ShImmediate commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get() → EventOnce

<pre># SCPI: SYSTem:SHIMmediate value: enums.EventOnce = driver.system.shImmediate.get()</pre>
--

Executes any received remote commands that cause changes to the hardware and have not been executed yet due to a SYST:SHIM:STAT OFF command.

return

hw_update: No help available

set(hw_update: EventOnce) → None

```
# SCPI: SYSTem:SHIMmediate
driver.system.shImmediate.set(hw_update = enums.EventOnce.ONCE)
```

Executes any received remote commands that cause changes to the hardware and have not been executed yet due to a SYST:SHIM:STAT OFF command.

param hw_update
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.shImmediate.clone()
```

Subgroups

6.20.26.1 State

SCPI Commands

```
SYSTem:SHIMmediate:STATe
```

class StateCls

State commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:SHIMmediate:STATe
value: bool = driver.system.shImmediate.state.get()
```

Determines when the remote commands that change hardware settings on the R&S FSWP are executed. Regardless of this setting, the firmware automatically sets the hardware when a sweep is started. This setting is not changed by the preset function.

return
state: ON | OFF | 0 | 1 OFF | 0 Remote commands that cause changes to the hardware are only executed when the SYSTem:SHIMmediate ONCE command is executed. ON | 1 Remote commands are always executed immediately when they are received by the instrument.

set(state: bool) → None

```
# SCPI: SYSTem:SHIMmediate:STATe
driver.system.shImmediate.state.set(state = False)
```

Determines when the remote commands that change hardware settings on the R&S FSWP are executed. Regardless of this setting, the firmware automatically sets the hardware when a sweep is started. This setting is not changed by the preset function.

param state
ON | OFF | 0 | 1 OFF | 0 Remote commands that cause changes to the hardware are only executed when the SYSTem:SHIMmediate ONCE command is executed. ON |

1 Remote commands are always executed immediately when they are received by the instrument.

6.20.27 Shutdown

SCPI Commands

```
SYSTem:SHUTdown
```

class ShutdownCls

Shutdown commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:SHUTdown
driver.system.shutdown.set()
```

This command shuts down the instrument.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:SHUTdown
driver.system.shutdown.set_with_opc()
```

This command shuts down the instrument.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.28 Srecorder

class SrecorderCls

Srecorder commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.srecorder.clone()
```

Subgroups

6.20.28.1 Sync

SCPI Commands

SYSTem:SREcorder:SYNC

class SyncCls

Sync commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → bool

```
# SCPI: SYSTem:SREcorder:SYNC
value: bool = driver.system.srecorder.sync.get()
```

No command help available

return

state: No help available

set(state: bool) → None

```
# SCPI: SYSTem:SREcorder:SYNC
driver.system.srecorder.sync.set(state = False)
```

No command help available

param state

No help available

6.20.29 Test

class TestCls

Test commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.test.clone()
```

Subgroups

6.20.29.1 Redo

SCPI Commands

SYSTem:TEST:REDO

class RedoCls

Redo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:TEST:REDO
driver.system.test.redo.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:TEST:REDO
driver.system.test.redo.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.20.29.2 Undo

SCPI Commands

```
SYSTem:TEST:UNDO
```

class UndoCls

Undo commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: SYSTem:TEST:UNDO
driver.system.test.undo.set()
```

No command help available

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: SYSTem:TEST:UNDO
driver.system.test.undo.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.21 Trace<Window>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.trace.repcap_window_get()
driver.trace.repcap_window_set(repcap.Window.Nr1)
```

SCPI Commands

```
TRACe<Window>:COPY
```

class TraceCls

Trace commands group definition. 11 total commands, 2 Subgroups, 1 group commands Repeated Capability: Window, default value after init: Window.Nr1

copy(target_trace: TraceNumber, source_trace: TraceNumber, window=Window.Default) → None

```
# SCPI: TRACe<n>:COPY
driver.trace.copy(target_trace = enums.TraceNumber.BTOBits, source_trace =
↳enums.TraceNumber.BTOBits, window = repcap.Window.Default)
```

This command copies data from one trace to another.

param target_trace
No help available

param source_trace
No help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.clone()
```

Subgroups

6.21.1 Data

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA
```

class DataCls

Data commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get(*trace_type*: *TraceNumber*, *window*=*Window.Default*) → List[float]

```
# SCPI: TRACe<n>[:DATA]
value: List[float] = driver.trace.data.get(trace_type = enums.TraceNumber.
↳BTObits, window = repcap.Window.Default)
```

This command queries current trace data and measurement results. The data format depends on method RsFswp.FormatPy.Data. set.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_ydata: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.data.clone()
```

Subgroups

6.21.1.1 Memory

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA:MEMory
```

class MemoryCls

Memory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*trace_type*: *TraceNumber*, *points_offset*: *int*, *points_count*: *int*, *window*=*Window.Default*) → List[float]

```
# SCPI: TRACe<n>[:DATA]:MEMory
value: List[float] = driver.trace.data.memory.get(trace_type = enums.
↳TraceNumber.BTObits, points_offset = 1, points_count = 1, window = repcap.
↳Window.Default)
```

This command queries the previously captured trace data for the specified trace from the memory. As an offset and number of sweep points to be retrieved can be specified, the trace data can be retrieved in smaller portions, making the command faster than the TRAC:DATA? command. This is useful if only specific parts of the trace data are of interest. If no parameters are specified with the command, the entire trace data is retrieved; in this case, the command returns the same results as TRAC:DATA? TRACE1.

param trace_type

No help available

param points_offset

No help available

param points_count

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_ydata: No help available

6.21.1.2 X

SCPI Commands

```
FORMAT REAL,32;TRACe<Window>:DATA:X
```

class XCls

X commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(trace_type: TraceNumber, window=Window.Default) → List[float]

```
# SCPI: TRACe<n>[:DATA]:X
value: List[float] = driver.trace.data.x.get(trace_type = enums.TraceNumber.
↳BTOBits, window = repcap.Window.Default)
```

This command queries the horizontal trace data for each sweep point in the specified window, for example the frequency in frequency domain or the time in time domain measurements. This is especially useful for traces with non-equidistant x-values.

param trace_type

No help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

trace_xdata: No help available

6.21.2 Iq

class IqCls

Iq commands group definition. 7 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.clone()
```

Subgroups

6.21.2.1 Data

SCPI Commands

```
FORMAT REAL,32;TRACe:IQ:DATA
```

class DataCls

Data commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: TRACe:IQ:DATA
value: List[float] = driver.trace.iq.data.get()
```

This command initiates a measurement with the current settings and returns the captured data from I/Q measurements. This command corresponds to: INIT:IMM;*WAI;method **RsFswp.Trace.Iq.Data.Memory.get_** However, the method **RsFswp.Trace.Iq.Data.get_** command is quicker in comparison. Note: Using the command with the *RST values for the TRACe:IQ:SET command, the following minimum buffer sizes for the response data are recommended: ASCII format 10 kBytes, binary format: 2 kBytes

return

iq_data: Measured voltage for I and Q component for each sample that has been captured during the measurement. The number of samples depends on TRACe:IQ:SET. In ASCII format, the number of results is 2* the number of samples. The data format depends on method **RsFswp.Applications.IqAnalyzer.Trace.Iq.Data.FormatPy.set**. Unit: V

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.data.clone()
```

Subgroups

6.21.2.1.1 Memory

SCPI Commands

```
FORMAT REAL,32;TRACe:IQ:DATA:MEMory
```

class MemoryCls

Memory commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(points_offset: int, points_count: int) → List[float]

```
# SCPI: TRACe:IQ:DATA:MEMory
value: List[float] = driver.trace.iq.data.memory.get(points_offset = 1, points_
↪count = 1)
```

This command queries the I/Q data currently stored in the capture buffer of the R&S FSWP. By default, the command returns all I/Q data in the memory. You can, however, narrow down the amount of data that the command returns using the optional parameters. If no parameters are specified with the command, the entire trace data is retrieved. In this case, the command returns the same results as method **RsFswp.Trace.Iq.Data.get_**. (Note, however, that the method `RsFswp.Trace.Iq. Data.get_` command initiates a new measurement before returning the captured values, rather than returning the existing data in the memory.) The command returns a comma-separated list of the measured values in floating point format (comma-separated values = CSV) . The number of values returned is 2 * the number of complex samples. The total number of complex samples is displayed in the channel bar in manual operation and can be calculated as: $\text{<SampleRate> * <CaptureTime>}$ (See `TRACe:IQ:SET`, method `RsFswp.Applications.IqAnalyzer.Trace.Iq.SymbolRate.set` and `[SENSe:]SWEep:TIME`)

param points_offset

No help available

param points_count

No help available

return

`iq_data`: Measured value pair (I,Q) for each sample that has been recorded. By default, the first half of the list contains the I values, the second half the Q values. The order can be configured using method `RsFswp.Applications.IqAnalyzer.Trace.Iq.Data.FormatPy.set`. The data format of the individual values depends on method `RsFswp.FormatPy.Data.set`. Unit: V

6.21.2.1.2 MemoryAll

SCPI Commands

```
FORMAT REAL, 32;TRACe:IQ:DATA:MEMory
```

class MemoryAllCls

MemoryAll commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → List[float]

```
# SCPI: TRACe:IQ:DATA:MEMory
value: List[float] = driver.trace.iq.data.memoryAll.get()
```

This command queries the I/Q data currently stored in the capture buffer of the R&S FSWP. By default, the command returns all I/Q data in the memory. You can, however, narrow down the amount of data that the command returns using the optional parameters. If no parameters are specified with the command, the entire trace data is retrieved. In this case, the command returns the same results as method **RsFswp.Trace.Iq.Data.get_**. (Note, however, that the method `RsFswp.Trace.Iq. Data.get_` command initiates a new measurement before returning the captured values, rather than returning the existing data in the memory.) The command returns a comma-separated list of the measured values in floating point format (comma-separated values = CSV) . The number of values returned is 2 * the number of complex samples. The total number of complex samples is displayed in the channel bar in manual operation and can be calculated as: $\text{<SampleRate> * <CaptureTime>}$ (See `TRACe:IQ:SET`, method `RsFswp.Applications.IqAnalyzer.Trace.Iq.SymbolRate.set` and `[SENSe:]SWEep:TIME`)

return

`iq_data`: Measured value pair (I,Q) for each sample that has been recorded. By default, the first half of the list contains the I values, the second

half the Q values. The order can be configured using method RsFswp.Applications.IqAnalyzer.Trace.Iq.Data.FormatPy.set. The data format of the individual values depends on method RsFswp.FormatPy.Data.set. Unit: V

6.21.2.2 File

class FileCls

File commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.file.clone()
```

Subgroups

6.21.2.2.1 Repetition

class RepetitionCls

Repetition commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.file.repetition.clone()
```

Subgroups

6.21.2.2.1.1 Count

SCPI Commands

```
TRACe:IQ:FILE:REPetition:COUNT
```

class CountCls

Count commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNT
value: float = driver.trace.iq.file.repetition.count.get()
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

```
return
    repetition_count: integer
```

set(*repetition_count: float*) → None

```
# SCPI: TRACe:IQ:FILE:REPetition:COUNT
driver.trace.iq.file.repetition.count.set(repetition_count = 1.0)
```

Determines how often the data stream is repeatedly copied in the I/Q data memory. If the available memory is not sufficient for the specified number of repetitions, the largest possible number of complete data streams is used.

param repetition_count
integer

6.21.2.3 Scapture

class ScaptureCls

Scapture commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.scapture.clone()
```

Subgroups

6.21.2.3.1 Boundary

SCPI Commands

```
TRACe<Window>:IQ:SCAPture:BOUNDary
```

class BoundaryCls

Boundary commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*window=Window.Default*) → List[int]

```
# SCPI: TRACe<n>:IQ:SCAPture:BOUNDary
value: List[int] = driver.trace.iq.scapture.boundary.get(window = repcap.Window.
↳Default)
```

No command help available

param window
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return
indices: No help available

6.21.2.3.2 Tstamp

class TstampCls

Tstamp commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.iq.scapture.tstamp.clone()
```

Subgroups

6.21.2.3.2.1 Sstart

SCPI Commands

```
TRACE<Window>:IQ:SCAPture:TSTamp:SSTart
```

class SstartCls

Sstart commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → List[float]

```
# SCPI: TRACE<n>:IQ:SCAPture:TSTamp:SSTart
value: List[float] = driver.trace.iq.scapture.tstamp.sstart.get(window = repcap.
↳ Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

timestamps: No help available

6.21.2.3.2.2 Trigger

SCPI Commands

```
TRACE<Window>:IQ:SCAPture:TSTamp:TRIGger
```

class TriggerCls

Trigger commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(window=Window.Default) → List[float]

```
# SCPI: TRACE<n>:IQ:SCAPture:TSTamp:TRIGger
value: List[float] = driver.trace.iq.scapture.tstamp.trigger.get(window =
↳ repcap.Window.Default)
```

No command help available

param window

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Trace')

return

timestamps: No help available

6.22 Trigger

class TriggerCls

Trigger commands group definition. 23 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.clone()
```

Subgroups

6.22.1 Iq

class IqCls

Iq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.iq.clone()
```

Subgroups

6.22.1.1 Master

class MasterCls

Master commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.iq.master.clone()
```

Subgroups

6.22.1.1.1 Source

SCPI Commands

```
TRIGger:IQ:MASTer:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerPort

```
# SCPI: TRIGger:IQ:MASTer:SOURce
value: enums.TriggerPort = driver.trigger.iq.master.source.get()
```

No command help available

```
return
    ms_trig_iq_master_source: No help available
```

set(dev_name: str, ms_trig_iq_master_source: TriggerPort) → None

```
# SCPI: TRIGger:IQ:MASTer:SOURce
driver.trigger.iq.master.source.set(dev_name = '1', ms_trig_iq_master_source =
↳enums.TriggerPort.EXT1)
```

No command help available

```
param dev_name
    No help available
```

```
param ms_trig_iq_master_source
    No help available
```

6.22.1.2 Sender

class SenderCls

Sender commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.iq.sender.clone()
```

Subgroups

6.22.1.2.1 Source

SCPI Commands

```
TRIGger:IQ:SENDER:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerPort

```
# SCPI: TRIGger:IQ:SENDER:SOURce
value: enums.TriggerPort = driver.trigger.iq.sender.source.get()
```

No command help available

```
    return
        trigger_iq_source: No help available
```

set(dev_name: str, trigger_iq_source: TriggerPort) → None

```
# SCPI: TRIGger:IQ:SENDER:SOURce
driver.trigger.iq.sender.source.set(dev_name = '1', trigger_iq_source = enums.
    ↪TriggerPort.EXT1)
```

No command help available

```
    param dev_name
        No help available
```

```
    param trigger_iq_source
        No help available
```

6.22.2 Master

class MasterCls

Master commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.master.clone()
```

Subgroups

6.22.2.1 Port

SCPI Commands

```
TRIGger:MASTer:PORT
```

class PortCls

Port commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerPort

```
# SCPI: TRIGger:MASTer:PORT
value: enums.TriggerPort = driver.trigger.master.port.get()
```

No command help available

return

ms_master_port: No help available

set(dev_name: str, ms_master_port: TriggerPort) → None

```
# SCPI: TRIGger:MASTer:PORT
driver.trigger.master.port.set(dev_name = '1', ms_master_port = enums.
↳ TriggerPort.EXT1)
```

No command help available

param dev_name

No help available

param ms_master_port

No help available

6.22.3 Sender

class SenderCls

Sender commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sender.clone()
```

Subgroups

6.22.3.1 Port

SCPI Commands

```
TRIGger:SENDER:PORT
```

class PortCls

Port commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerPort

```
# SCPI: TRIGger:SENDER:PORT
value: enums.TriggerPort = driver.trigger.sender.port.get()
```

No command help available

return
port: No help available

set(dev_name: str, port: TriggerPort) → None

```
# SCPI: TRIGger:SENDER:PORT
driver.trigger.sender.port.set(dev_name = '1', port = enums.TriggerPort.EXT1)
```

No command help available

param dev_name
No help available

param port
No help available

6.22.4 Sequence

class SequenceCls

Sequence commands group definition. 19 total commands, 10 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.clone()
```

Subgroups

6.22.4.1 BbPower

class BbPowerCls

BbPower commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.bbPower.clone()
```

Subgroups

6.22.4.1.1 Holdoff

SCPI Commands

```
TRIGger:SEquence:BBPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:BBPower:HOLDoff
value: float = driver.trigger.sequence.bbPower.holdoff.get()
```

This command defines the holding time before the baseband power trigger event. Note that this command is maintained for compatibility reasons only. Use the method RsFswp.Applications.K50_Spurious.Trigger.Sequence.IfPower.Holdoff.set command for new remote control programs.

return

period: Range: 150 ns to 1000 s, Unit: S

set(period: float) → None

```
# SCPI: TRIGger[:SEquence]:BBPower:HOLDoff
driver.trigger.sequence.bbPower.holdoff.set(period = 1.0)
```

This command defines the holding time before the baseband power trigger event. Note that this command is maintained for compatibility reasons only. Use the method RsFswp.Applications.K50_Spurious.Trigger.Sequence.IfPower.Holdoff.set command for new remote control programs.

param period

Range: 150 ns to 1000 s, Unit: S

6.22.4.2 Dtime**SCPI Commands**

TRIGger:SEquence:DTIME

class DtimeCls

Dtime commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:DTIME
value: float = driver.trigger.sequence.dtime.get()
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

return

dropout_time: Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

set(dropout_time: float) → None

```
# SCPI: TRIGger[:SEquence]:DTIME
driver.trigger.sequence.dtime.set(dropout_time = 1.0)
```

Defines the time the input signal must stay below the trigger level before a trigger is detected again.

param dropout_time

Dropout time of the trigger. Range: 0 s to 10.0 s , Unit: S

6.22.4.3 Holdoff**class HoldoffCls**

Holdoff commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.holdoff.clone()
```

Subgroups**6.22.4.3.1 Time****SCPI Commands**

TRIGger:SEquence:HOLDoff:TIME

class TimeCls

Time commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
value: float = driver.trigger.sequence.holdoff.time.get()
```

Defines the time offset between the trigger event and the start of the measurement.

return

offset: For measurements in the frequency domain, the range is 0 s to 30 s. For measurements in the time domain, the range is the negative measurement time to 30 s. Unit: S

set(offset: float) → None

```
# SCPI: TRIGger[:SEquence]:HOLDoff[:TIME]
driver.trigger.sequence.holdoff.time.set(offset = 1.0)
```

Defines the time offset between the trigger event and the start of the measurement.

param offset

For measurements in the frequency domain, the range is 0 s to 30 s. For measurements in the time domain, the range is the negative measurement time to 30 s. Unit: S

6.22.4.4 IfPower**class IfPowerCls**

IfPower commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.ifPower.clone()
```

Subgroups**6.22.4.4.1 Holdoff****SCPI Commands**

```
TRIGger:SEquence:IFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
value: float = driver.trigger.sequence.ifPower.holdoff.get()
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) . Note: If you perform gated measurements in combination with the IF Power trigger, the R&S FSWP ignores the holding time for frequency sweep, FFT sweep, zero span and I/Q data measurements.

return

period: Range: 0 s to 10 s, Unit: S

set(*period: float*) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HOLDoff
driver.trigger.sequence.ifPower.holdoff.set(period = 1.0)
```

This command defines the holding time before the next trigger event. Note that this command can be used for any trigger source, not just IF Power (despite the legacy keyword) . Note: If you perform gated measurements in combination with the IF Power trigger, the R&S FSWP ignores the holding time for frequency sweep, FFT sweep, zero span and I/Q data measurements.

param period

Range: 0 s to 10 s, Unit: S

6.22.4.4.2 Hysteresis

SCPI Commands

```
TRIGger:SEquence:IFPower:HYSTeresis
```

class HysteresisCls

Hysteresis commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
value: float = driver.trigger.sequence.ifPower.hysteresis.get()
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

return

hysteresis: Range: 3 dB to 50 dB, Unit: DB

set(*hysteresis: float*) → None

```
# SCPI: TRIGger[:SEquence]:IFPower:HYSTeresis
driver.trigger.sequence.ifPower.hysteresis.set(hysteresis = 1.0)
```

This command defines the trigger hysteresis, which is only available for 'IF Power' trigger sources.

param hysteresis

Range: 3 dB to 50 dB, Unit: DB

6.22.4.5 Level

class LevelCls

Level commands group definition. 9 total commands, 8 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.level.clone()
```

Subgroups

6.22.4.5.1 Am

class AmCls

Am commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.level.am.clone()
```

Subgroups

6.22.4.5.1.1 Absolute

SCPI Commands

```
TRIGger:SEquence:LEVel:AM:ABSolute
```

class AbsoluteCls

Absolute commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:AM[:ABSolute]
value: float = driver.trigger.sequence.level.am.absolute.get()
```

The command sets the level when RF power signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

return

level: Range: -100 to +30, Unit: dBm

set(level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:AM[:ABSolute]
driver.trigger.sequence.level.am.absolute.set(level = 1.0)
```

The command sets the level when RF power signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

param level

Range: -100 to +30, Unit: dBm

6.22.4.5.1.2 Relative

SCPI Commands

TRIGger:SEquence:LEVel:AM:RELative

class RelativeCls

Relative commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

<pre># SCPI: TRIGger[:SEquence]:LEVel:AM:RELative value: float = driver.trigger.sequence.level.am.relative.get()</pre>
--

The command sets the level when AM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

return

level: Range: -100 to +100, Unit: %

set(level: float) → None

<pre># SCPI: TRIGger[:SEquence]:LEVel:AM:RELative driver.trigger.sequence.level.am.relative.set(level = 1.0)</pre>
--

The command sets the level when AM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

param level

Range: -100 to +100, Unit: %

6.22.4.5.2 External<ExternalPort>

RepCap Settings

<pre># Range: Nr1 .. Nr3 rc = driver.trigger.sequence.level.external.repcap_externalPort_get() driver.trigger.sequence.level.external.repcap_externalPort_set(repcap.ExternalPort.Nr1)</pre>
--

SCPI Commands

```
TRIGger:SEquence:LEVel:EXTeRnal<ExternalPort>
```

class ExternalCls

External commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ExternalPort, default value after init: ExternalPort.Nr1

get(*externalPort*=*ExternalPort.Default*) → float

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXTeRnal<tp>]
value: float = driver.trigger.sequence.level.external.get(externalPort = repcap.
↳ ExternalPort.Default)
```

This command defines the external trigger level.

INTRO_CMD_HELP: Prerequisites for this command

- Select the external trigger (method RsFswp.Applications.K50_Spurious.Trigger.Sequence.Source.set)
-

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

return

trigger_level: No help available

set(*trigger_level*: float, *externalPort*=*ExternalPort.Default*) → None

```
# SCPI: TRIGger[:SEquence]:LEVel[:EXTeRnal<tp>]
driver.trigger.sequence.level.external.set(trigger_level = 1.0, externalPort =
↳ repcap.ExternalPort.Default)
```

This command defines the external trigger level.

INTRO_CMD_HELP: Prerequisites for this command

- Select the external trigger (method RsFswp.Applications.K50_Spurious.Trigger.Sequence.Source.set)
-

param trigger_level

No help available

param externalPort

optional repeated capability selector. Default value: Nr1 (settable in the interface ‘External’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.level.external.clone()
```

6.22.4.5.3 Fm

SCPI Commands

```
TRIGger:SEquence:LEVel:FM
```

class FmCls

Fm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:FM
value: float = driver.trigger.sequence.level.fm.get()
```

The command sets the level when FM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

return
level: Range: -10 to +10, Unit: MHz

set(level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:FM
driver.trigger.sequence.level.fm.set(level = 1.0)
```

The command sets the level when FM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

param level
Range: -10 to +10, Unit: MHz

6.22.4.5.4 IfPower

SCPI Commands

```
TRIGger:SEquence:LEVel:IFPower
```

class IfPowerCls

IfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
value: float = driver.trigger.sequence.level.ifPower.get()
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

return

trigger_level: For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

set(trigger_level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IFPower
driver.trigger.sequence.level.ifPower.set(trigger_level = 1.0)
```

This command defines the power level at the third intermediate frequency that must be exceeded to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param trigger_level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.22.4.5.5 IqPower**SCPI Commands**

```
TRIGger:SEquence:LEVel:IQPower
```

class IqPowerCls

IqPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
value: float = driver.trigger.sequence.level.iqPower.get()
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

return

trigger_level: Range: -130 dBm to 30 dBm, Unit: DBM

set(trigger_level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:IQPower
driver.trigger.sequence.level.iqPower.set(trigger_level = 1.0)
```

This command defines the magnitude the I/Q data must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered.

param trigger_level

Range: -130 dBm to 30 dBm, Unit: DBM

6.22.4.5.6 Pm

SCPI Commands

`TRIGger:SEquence:LEVel:PM`

class PmCls

Pm commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:PM
value: float = driver.trigger.sequence.level.pm.get()
```

The command sets the level when PM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

return

level: Range: -1000 to +1000, Unit: RAD | DEG

set(level: float) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:PM
driver.trigger.sequence.level.pm.set(level = 1.0)
```

The command sets the level when PM-modulated signals are used as trigger source. For triggering to be successful, the measurement time must cover at least 5 periods of the audio signal.

param level

Range: -1000 to +1000, Unit: RAD | DEG

6.22.4.5.7 RfPower

SCPI Commands

`TRIGger:SEquence:LEVel:RfPower`

class RfPowerCls

RfPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:RfPower
value: float = driver.trigger.sequence.level.rfPower.get()
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

return

trigger_level: For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

set(*trigger_level: float*) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:RFPower
driver.trigger.sequence.level.rfPower.set(trigger_level = 1.0)
```

This command defines the power level the RF input must exceed to cause a trigger event. Note that any RF attenuation or preamplification is considered when the trigger level is analyzed. If defined, a reference level offset is also considered. The input signal must be between 500 MHz and 8 GHz.

param trigger_level

For details on available trigger levels and trigger bandwidths, see the data sheet. Unit: DBM

6.22.4.5.8 Video

SCPI Commands

```
TRIGger:SEquence:LEVel:VIDeo
```

class VideoCls

Video commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:LEVel:VIDeo
value: float = driver.trigger.sequence.level.video.get()
```

No command help available

return

level: No help available

set(*level: float*) → None

```
# SCPI: TRIGger[:SEquence]:LEVel:VIDeo
driver.trigger.sequence.level.video.set(level = 1.0)
```

No command help available

param level

No help available

6.22.4.6 Oscilloscope

class OscilloscopeCls

Oscilloscope commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.oscilloscope.clone()
```

Subgroups

6.22.4.6.1 Coupling

SCPI Commands

```
TRIGger:SEquence:OSCilloscope:COUpling
```

class CouplingCls

Coupling commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → CouplingTypeB

```
# SCPI: TRIGger[:SEquence]:OSCilloscope:COUpling
value: enums.CouplingTypeB = driver.trigger.sequence.oscilloscope.coupling.get()
```

No command help available

return

coupling_type: No help available

set(coupling_type: CouplingTypeB) → None

```
# SCPI: TRIGger[:SEquence]:OSCilloscope:COUpling
driver.trigger.sequence.oscilloscope.coupling.set(coupling_type = enums.
↪CouplingTypeB.AC)
```

No command help available

param coupling_type

No help available

6.22.4.7 RfPower

class RfPowerCls

RfPower commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.rfPower.clone()
```


Subgroups

6.22.4.7.1 Holdoff

SCPI Commands

```
TRIGger:SEquence:RFPower:HOLDoff
```

class HoldoffCls

Holdoff commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:RFPower:HOLDoff
value: float = driver.trigger.sequence.rfPower.holdoff.get()
```

No command help available

return
time: No help available

set(time: float) → None

```
# SCPI: TRIGger[:SEquence]:RFPower:HOLDoff
driver.trigger.sequence.rfPower.holdoff.set(time = 1.0)
```

No command help available

param time
No help available

6.22.4.8 Slope

SCPI Commands

```
TRIGger:SEquence:SLOPe
```

class SlopeCls

Slope commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → SlopeType

```
# SCPI: TRIGger[:SEquence]:SLOPe
value: enums.SlopeType = driver.trigger.sequence.slope.get()
```

This command selects the trigger slope.

return
slope: No help available

set(slope: SlopeType) → None

```
# SCPI: TRIGger[:SEquence]:SLOPe
driver.trigger.sequence.slope.set(slope = enums.SlopeType.NEGative)
```

This command selects the trigger slope.

param slope

POSitive | NEGative POSitive Triggers when the signal rises to the trigger level (rising edge) . NEGative Triggers when the signal drops to the trigger level (falling edge) .

6.22.4.9 Source

SCPI Commands

```
TRIGger:SEquence:SOURce
```

class SourceCls

Source commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → TriggerSeqSource

```
# SCPI: TRIGger[:SEquence]:SOURce
value: enums.TriggerSeqSource = driver.trigger.sequence.source.get()
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

return

source: IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’. RFPower First intermediate frequency (Frequency and time domain measurements only.) IF-Power Second intermediate frequency IQPower Magnitude of sampled I/Q data For applications that process I/Q data, such as the I/Q Analyzer or optional applications. BBPower Baseband power (for digital input via the optional ‘Digital Baseband’ interface PSEN External power sensor AF AF power signal FM FM power signal AM corresponds to the RF power signal AMRelative corresponds to the AM signal PM PM power signal

set(source: TriggerSeqSource) → None

```
# SCPI: TRIGger[:SEquence]:SOURce
driver.trigger.sequence.source.set(source = enums.TriggerSeqSource.ACVideo)
```

This command selects the trigger source. Note on external triggers: If a measurement is configured to wait for an external trigger signal in a remote control program, remote control is blocked until the trigger is received and the program can continue. Make sure that this situation is avoided in your remote control programs.

param source

IMMEDIATE Free Run EXT | EXT2 Trigger signal from one of the ‘Trigger Input/Output’ connectors. Note: Connector must be configured for ‘Input’. RFPower First intermediate frequency (Frequency and time domain measurements only.) IF-Power Second intermediate frequency IQPower Magnitude of sampled I/Q data For applications that process I/Q data, such as the I/Q Analyzer or optional applications. BBPower Baseband power (for digital input via the optional ‘Digital Baseband’ interface PSEN External power sensor AF AF power signal FM FM power signal AM

corresponds to the RF power signal AMRelative corresponds to the AM signal PM PM
power signal

6.22.4.10 Time

class TimeCls

Time commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.sequence.time.clone()
```

Subgroups

6.22.4.10.1 Rinterval

SCPI Commands

```
TRIGger:SEquence:TIME:RINTerval
```

class RintervalCls

Interval commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → float

```
# SCPI: TRIGger[:SEquence]:TIME:RINTerval
value: float = driver.trigger.sequence.time.rinterval.get()
```

This command defines the repetition interval for the time trigger.

return

interval: numeric value Range: 2 ms to 5000 s, Unit: S

set(interval: float) → None

```
# SCPI: TRIGger[:SEquence]:TIME:RINTerval
driver.trigger.sequence.time.rinterval.set(interval = 1.0)
```

This command defines the repetition interval for the time trigger.

param interval

numeric value Range: 2 ms to 5000 s, Unit: S

6.23 TriggerInvoke

SCPI Commands

`*TRG`

class TriggerInvokeCls

TriggerInvoke commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: *TRG
driver.triggerInvoke.set()
```

Trigger Triggers all actions waiting for a trigger event. In particular, **TRG* generates a manual trigger signal. This common command complements the commands of the TRIGger subsystem. **TRG* corresponds to the INITiate:IMMediate command.

set_with_opc(opc_timeout_ms: int = -1) → None

```
# SCPI: *TRG
driver.triggerInvoke.set_with_opc()
```

Trigger Triggers all actions waiting for a trigger event. In particular, **TRG* generates a manual trigger signal. This common command complements the commands of the TRIGger subsystem. **TRG* corresponds to the INITiate:IMMediate command.

Same as set, but waits for the operation to complete before continuing further. Use the RsFswp.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms

Maximum time to wait in milliseconds, valid only for this call.

6.24 Unit

class UnitCls

Unit commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.unit.clone()
```

Subgroups

6.24.1 Angle

SCPI Commands

UNIT:ANGLE

class AngleCls

Angle commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → AngleUnit

```
# SCPI: UNIT:ANGLE
value: enums.AngleUnit = driver.unit.angle.get()
```

This command selects the unit for angles (for PM display, <n> is irrelevant) . This command is identical to method RsFswp. Calculate.Unit.Angle.set

return
unit: DEG | RAD

set(unit: AngleUnit) → None

```
# SCPI: UNIT:ANGLE
driver.unit.angle.set(unit = enums.AngleUnit.DEG)
```

This command selects the unit for angles (for PM display, <n> is irrelevant) . This command is identical to method RsFswp. Calculate.Unit.Angle.set

param unit
DEG | RAD

6.24.2 Pmeter<PowerMeter>

RepCap Settings

```
# Range: Nr1 .. Nr16
rc = driver.unit.pmeter.repcap_powerMeter_get()
driver.unit.pmeter.repcap_powerMeter_set(repcap.PowerMeter.Nr1)
```

class PmeterCls

Pmeter commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: PowerMeter, default value after init: PowerMeter.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.unit.pmeter.clone()
```

Subgroups

6.24.2.1 Power

SCPI Commands

```
UNIT:PMETer<PowerMeter>:POWer
```

class PowerCls

Power commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → PowerMeterUnit

```
# SCPI: UNIT:PMETer<p>:POWer
value: enums.PowerMeterUnit = driver.unit.pmeter.power.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

unit: No help available

set(*unit: PowerMeterUnit, powerMeter=PowerMeter.Default*) → None

```
# SCPI: UNIT:PMETer<p>:POWer
driver.unit.pmeter.power.set(unit = enums.PowerMeterUnit.DBM, powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param unit

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.unit.pmeter.power.clone()
```

Subgroups

6.24.2.1.1 Ratio

SCPI Commands

```
UNIT:PMETer<PowerMeter>:POWer:RATio
```

class RatioCls

Ratio commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*powerMeter=PowerMeter.Default*) → UnitMode

```
# SCPI: UNIT:PMETer<p>:POWer:RATio
value: enums.UnitMode = driver.unit.pmeter.power.ratio.get(powerMeter = repcap.
↳PowerMeter.Default)
```

No command help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

return

unit: No help available

set(*unit: UnitMode, powerMeter=PowerMeter.Default*) → None

```
# SCPI: UNIT:PMETer<p>:POWer:RATio
driver.unit.pmeter.power.ratio.set(unit = enums.UnitMode.DB, powerMeter =
↳repcap.PowerMeter.Default)
```

No command help available

param unit

No help available

param powerMeter

optional repeated capability selector. Default value: Nr1 (settable in the interface 'Pmeter')

6.24.3 Thd

SCPI Commands

UNIT:THD

class ThdCls

Thd commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get() → UnitMode

<pre># SCPI: UNIT:THD value: enums.UnitMode = driver.unit.thd.get()</pre>

Selects the unit for THD measurements (<n> is irrelevant) . This command is identical to CALC:UNIT:THD

return
mode: DB | PCT

set(mode: UnitMode) → None

<pre># SCPI: UNIT:THD driver.unit.thd.set(mode = enums.UnitMode.DB)</pre>

Selects the unit for THD measurements (<n> is irrelevant) . This command is identical to CALC:UNIT:THD

param mode
DB | PCT

RSFSWP UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsFswp.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see [here](#)

Access snippet: `logger = RsFswp.utilities.logger`

property driver_version: `str`

Returns the instrument driver version.

property idn_string: `str`

Returns instrument's identification string - the response on the SCPI command `*IDN?`

property manufacturer: `str`

Returns manufacturer of the instrument.

property full_instrument_model_name: `str`

Returns the current instrument's full name e.g. 'FSW26'.

property instrument_model_name: `str`

Returns the current instrument's family name e.g. 'FSW'.

property supported_models: `List[str]`

Returns a list of the instrument models supported by this instrument driver.

property instrument_firmware_version: `str`

Returns instrument's firmware version.

property instrument_serial_number: `str`

Returns instrument's serial_number.

query_opc(*timeout: int = 0*) → `int`

SCPI command: `*OPC?` Queries the instrument's OPC bit and hence it waits until the instrument reports operation complete. If you define `timeout > 0`, the VISA timeout is set to that value just for this method call.

property instrument_status_checking: `bool`

Sets / returns Instrument Status Checking. When True (default is True), all the driver methods and properties are sending "SYSTem:ERRor?" at the end to immediately react on error that might have occurred. We recommend to keep the state checking ON all the time. Switch it OFF only in rare cases when you require maximum speed. The default state after initializing the session is ON.

property encoding: str

Returns string<=>bytes encoding of the session.

property opc_query_after_write: bool

Sets / returns Instrument *OPC? query sending after each command write. When True, (default is False) the driver sends *OPC? every time a write command is performed. Use this if you want to make sure your sequence is performed command-after-command.

property bin_float_numbers_format: BinFloatFormat

Sets / returns format of float numbers when transferred as binary data.

property bin_int_numbers_format: BinIntFormat

Sets / returns format of integer numbers when transferred as binary data.

clear_status() → None

Clears instrument's status system, the session's I/O buffers and the instrument's error queue.

query_all_errors() → List[str]

Queries and clears all the errors from the instrument's error queue. The method returns list of strings as error messages. If no error is detected, the return value is None. The process is: querying 'SYS-Tem:ERror?' in a loop until the error queue is empty. If you want to include the error codes, call the query_all_errors_with_codes()

query_all_errors_with_codes() → List[Tuple[int, str]]

Queries and clears all the errors from the instrument's error queue. The method returns list of tuples (code: int, message: str). If no error is detected, the return value is None. The process is: querying 'SYS-Tem:ERror?' in a loop until the error queue is empty.

property instrument_options: List[str]

Returns all the instrument options. The options are sorted in the ascending order starting with K-options and continuing with B-options.

reset() → None

SCPI command: *RST Sends *RST command + calls the clear_status().

default_instrument_setup() → None

Custom steps performed at the init and at the reset().

self_test(timeout: Optional[int] = None) → Tuple[int, str]

SCPI command: *TST? Performs instrument's self-test. Returns tuple (code:int, message: str). Code 0 means the self-test passed. You can define the custom timeout in milliseconds. If you do not define it, the default selftest timeout is used (usually 60 secs).

is_connection_active() → bool

Returns true, if the VISA connection is active and the communication with the instrument still works.

reconnect(force_close: bool = False) → bool

If the connection is not active, the method tries to reconnect to the device. If the connection is active, and force_close is False, the method does nothing. If the connection is active, and force_close is True, the method closes, and opens the session again. Returns True, if the reconnection has been performed.

property resource_name: int

Returns the resource name used in the constructor

property opc_timeout: int

Sets / returns timeout in milliseconds for all the operations that use OPC synchronization.

property visa_timeout: int

Sets / returns visa IO timeout in milliseconds.

property data_chunk_size: int

Sets / returns the maximum size of one block transferred during write/read operations

property visa_manufacturer: int

Returns the manufacturer of the current VISA session.

process_all_commands() → None

SCPI command: **WAI* Stops further commands processing until all commands sent before **WAI* have been executed.

write_str(cmd: str) → None

Writes the command to the instrument.

write(cmd: str) → None

This method is an alias to the write_str(). Writes the command to the instrument as string.

write_int(cmd: str, param: int) → None

Writes the command to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2'

write_int_with_opc(cmd: str, param: int, timeout: Optional[int] = None) → None

Writes the command with OPC to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2' If you do not provide timeout, the method uses current opc_timeout.

write_float(cmd: str, param: float) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6'

write_float_with_opc(cmd: str, param: float, timeout: Optional[int] = None) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6' If you do not provide timeout, the method uses current opc_timeout.

write_bool(cmd: str, param: bool) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON'

write_bool_with_opc(cmd: str, param: bool, timeout: Optional[int] = None) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON' If you do not provide timeout, the method uses current opc_timeout.

query_str(query: str) → str

Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query(query: str) → str

This method is an alias to the query_str(). Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query_bool(query: str) → bool

Sends the query to the instrument and returns the response as boolean.

query_int(*query: str*) → int

Sends the query to the instrument and returns the response as integer.

query_float(*query: str*) → float

Sends the query to the instrument and returns the response as float.

write_str_with_opc(*cmd: str, timeout: Optional[int] = None*) → None

Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current `opc_timeout`.

write_with_opc(*cmd: str, timeout: Optional[int] = None*) → None

This method is an alias to the `write_str_with_opc()`. Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current `opc_timeout`.

query_str_with_opc(*query: str, timeout: Optional[int] = None*) → str

Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current `opc_timeout`.

query_with_opc(*query: str, timeout: Optional[int] = None*) → str

This method is an alias to the `query_str_with_opc()`. Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current `opc_timeout`.

query_bool_with_opc(*query: str, timeout: Optional[int] = None*) → bool

Sends the opc-synced query to the instrument and returns the response as boolean. If you do not provide timeout, the method uses current `opc_timeout`.

query_int_with_opc(*query: str, timeout: Optional[int] = None*) → int

Sends the opc-synced query to the instrument and returns the response as integer. If you do not provide timeout, the method uses current `opc_timeout`.

query_float_with_opc(*query: str, timeout: Optional[int] = None*) → float

Sends the opc-synced query to the instrument and returns the response as float. If you do not provide timeout, the method uses current `opc_timeout`.

write_bin_block(*cmd: str, payload: bytes*) → None

Writes all the payload as binary data block to the instrument. The binary data header is added at the beginning of the transmission automatically, do not include it in the payload!!!

query_bin_block(*query: str*) → bytes

Queries binary data block to bytes. Throws an exception if the returned data was not a binary data. Returns `data:bytes`

query_bin_block_with_opc(*query: str, timeout: Optional[int] = None*) → bytes

Sends a OPC-synced query and returns binary data block to bytes. If you do not provide timeout, the method uses current `opc_timeout`.

query_bin_or_ascii_float_list(*query: str*) → List[float]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property `BinFloatFormat`, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_float_list_with_opc(*query: str, timeout: Optional[int] = None*) → List[float]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property `BinFloatFormat`, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current `opc_timeout`.

query_bin_or_ascii_int_list(query: str) → List[int]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_int_list_with_opc(query: str, timeout: Optional[int] = None) → List[int]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_block_to_file(query: str, file_path: str, append: bool = False) → None

Queries binary data block to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exist, it is created. Throws an exception if the returned data was not a binary data. Example for transferring a file from Instrument -> PC: query = f"MMEM:DATA? '{INSTR_FILE_PATH}'". Alternatively, use the dedicated methods for this purpose:

- send_file_from_pc_to_instrument()
- read_file_from_instrument_to_pc()

query_bin_block_to_file_with_opc(query: str, file_path: str, append: bool = False, timeout: Optional[int] = None) → None

Sends a OPC-synced query and writes the returned data to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exist, it is created. Throws an exception if the returned data was not a binary data.

write_bin_block_from_file(cmd: str, file_path: str) → None

Writes data from the file as binary data block to the instrument using the provided command. Example for transferring a file from PC -> Instrument: cmd = f"MMEM:DATA '{INSTR_FILE_PATH}',". Alternatively, use the dedicated methods for this purpose:

- send_file_from_pc_to_instrument()
- read_file_from_instrument_to_pc()

send_file_from_pc_to_instrument(source_pc_file: str, target_instr_file: str) → None

SCPI Command: MMEM:DATA

Sends file from PC to the instrument

read_file_from_instrument_to_pc(source_instr_file: str, target_pc_file: str, append_to_pc_file: bool = False) → None

SCPI Command: MMEM:DATA?

Reads file from instrument to the PC.

Set the append_to_pc_file to True if you want to append the read content to the end of the existing PC file

get_last_sent_cmd() → str

Returns the last commands sent to the instrument. Only works in simulation mode

get_lock() → RLock

Returns the thread lock for the current session.

By default:

- If you create standard new RsFswp instance with new VISA session, the session gets a new thread lock. You can assign it to other RsFswp sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsFswp from an existing session, the thread lock is shared automatically making both instances multi-thread safe.

You can always assign new thread lock by calling `driver.utilities.assign_lock()`

assign_lock(*lock: RLock*) → None

Assigns the provided thread lock.

clear_lock()

Clears the existing thread lock, making the current session thread-independent from others that might share the current thread lock.

sync_from(*source: Utilities*) → None

Synchronises these Utils with the source.

RSFSWP LOGGER

Check the usage in the Getting Started chapter [here](#).

class ScpiLogger

Base class for SCPI logging

mode

Sets the logging ON or OFF. Additionally, you can set the logging ON only for errors. Possible values:

- `LoggingMode.Off` - logging is switched OFF
- `LoggingMode.On` - logging is switched ON
- `LoggingMode.Errors` - logging is switched ON, but only for error entries
- `LoggingMode.Default` - sets the logging to default - the value you have set with `logger.default_mode`

default_mode

Sets / returns the default logging mode. You can recall the default mode by calling the `logger.mode = LoggingMode.Default`

Data Type

`LoggingMode`

device_name: str

Use this property to change the resource name in the log from the default Resource Name (e.g. `TCPIP::192.168.2.101::INSTR`) to another name e.g. `'MySigGen1'`.

set_logging_target(*target, console_log: Optional[bool] = None, udp_log: Optional[bool] = None*) → `None`

Sets logging target - the target must implement `write()` and `flush()`. You can optionally set the console and UDP logging ON or OFF. This method switches the logging target global OFF.

get_logging_target()

Based on the `global_mode`, it returns the logging target: either the local or the global one.

set_logging_target_global(*console_log: Optional[bool] = None, udp_log: Optional[bool] = None*) → `None`

Sets logging target to global. The global target must be defined. You can optionally set the console and UDP logging ON or OFF.

log_to_console

Returns logging to console status.

log_to_udp

Returns logging to UDP status.

log_to_console_and_udp

Returns true, if both logging to UDP and console in are True.

info_raw(log_entry: str, add_new_line: bool = True) → None

Method for logging the raw string without any formatting.

info(start_time: datetime, end_time: datetime, log_string_info: str, log_string: str) → None

Method for logging one info entry. For binary log_string, use the info_bin()

error(start_time: datetime, end_time: datetime, log_string_info: str, log_string: str) → None

Method for logging one error entry.

set_relative_timestamp(timestamp: datetime) → None

If set, the further timestamps will be relative to the entered time.

set_relative_timestamp_now() → None

Sets the relative timestamp to the current time.

get_relative_timestamp() → datetime

Based on the global_mode, it returns the relative timestamp: either the local or the global one.

clear_relative_timestamp() → None

Clears the reference time, and the further logging continues with absolute times.

flush() → None

Flush all the entries.

log_status_check_ok

Sets / returns the current status of status checking OK. If True (default), the log contains logging of the status checking 'Status check: OK'. If False, the 'Status check: OK' is skipped - the log is more compact. Errors will still be logged.

clear_cached_entries() → None

Clears potential cached log entries. Cached log entries are generated when the Logging is ON, but no target has been defined yet.

set_format_string(value: str, line_divider: str = '\n') → None

Sets new format string and line divider. If you just want to set the line divider, set the format string value=None. The original format string is: PAD_LEFT12(%START_TIME%) PAD_LEFT25(%DEVICE_NAME%) PAD_LEFT12(%DURATION%) %LOG_STRING_INFO%: %LOG_STRING%

restore_format_string() → None

Restores the original format string and the line divider to LF

abbreviated_max_len_ascii: int

Defines the maximum length of one ASCII log entry. Default value is 200 characters.

abbreviated_max_len_bin: int

Defines the maximum length of one Binary log entry. Default value is 2048 bytes.

abbreviated_max_len_list: int

Defines the maximum length of one list entry. Default value is 100 elements.

bin_line_block_size: int

Defines number of bytes to display in one line. Default value is 16 bytes.

udp_port

Returns udp logging port.

target_auto_flushing

Returns status of the auto-flushing for the logging target.

RSFSWP EVENTS

Check the usage in the Getting Started chapter [here](#).

class Events

Common Events class. Event-related methods and properties. Here you can set all the event handlers.

property before_query_handler: Callable

Returns the handler of before_query events.

Returns

current before_query_handler

property before_write_handler: Callable

Returns the handler of before_write events.

Returns

current before_write_handler

property io_events_include_data: bool

Returns the current state of the io_events_include_data See the setter for more details.

property on_read_handler: Callable

Returns the handler of on_read events.

Returns

current on_read_handler

property on_write_handler: Callable

Returns the handler of on_write events.

Returns

current on_write_handler

sync_from(source: Events) → None

Synchronises these Events with the source.

**CHAPTER
TEN**

INDEX

INDEX

Symbols

*TRG, 575, 728, 2276

A

abbreviated_max_len_ascii (*ScpiLogger attribute*), 2288
 abbreviated_max_len_bin (*ScpiLogger attribute*), 2288
 abbreviated_max_len_list (*ScpiLogger attribute*), 2288

B

bin_line_block_size (*ScpiLogger attribute*), 2288

C

CALCulate:SSEarch:TABLE:COLumn, 428
 CALCulate<Window>:BERate, 733
 CALCulate<Window>:CHRDetection:PNOise:FREQUENCY:START, 580
 CALCulate<Window>:CHRDetection:PNOise:FREQUENCY:STOP, 581
 CALCulate<Window>:CHRDetection:PNOise:LENGTH, 582
 CALCulate<Window>:CHRDetection:PNOise:OFFSET:BEGIN, 583
 CALCulate<Window>:CHRDetection:PNOise:OFFSET:END, 584
 CALCulate<Window>:CHRDetection:PNOise:REFERENCE, 584
 CALCulate<Window>:DDEM:BURSt:LENGTH, 734
 CALCulate<Window>:DDEM:SPECTrum:STaTe, 735
 CALCulate<Window>:DELTamarker:AOff, 1122
 CALCulate<Window>:DELTamarker<DeltaMarker>:AOff, 141, 392, 586, 737
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:AFPHase:REsult, 1123
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:AFPHase:STaTe, 1123
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:MODE, 1124
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:REsult, 1125
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:STaTe, 1126
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:BPOWer:STaTe:AFPHase:REsult, 1127
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:FIXed:STaTe, 1128
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:FIXed:STaTe:AFPHase:REsult, 1129
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:FIXed:STaTe:AFPHase:STaTe, 1130
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:FIXed:STaTe:AFPHase:STaTe:AFPHase:REsult, 1131
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:FIXed:STaTe:AFPHase:STaTe:AFPHase:STaTe:AFPHase:REsult, 1132
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:PNOise:STaTe, 1133
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:PNOise:STaTe:AFPHase:REsult, 1133
 CALCulate<Window>:DELTamarker<DeltaMarker>:FUNCTION:PNOise:STaTe:AFPHase:STaTe, 1134
 CALCulate<Window>:DELTamarker<DeltaMarker>:LINK:TO:MARKer, 393, 587, 1135
 CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:APEak, 737
 CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:LEFT, 395, 588, 738, 1136
 CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:NEXT, 142, 396, 589, 739, 1137
 CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:PEAK, 143, 396, 590, 739, 1137
 CALCulate<Window>:DELTamarker<DeltaMarker>:MAXimum:RIGHT, 397, 590, 740, 1138
 CALCulate<Window>:DELTamarker<DeltaMarker>:MBURSt:START, 740
 CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:LEFT, 397, 591, 741, 1139
 CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:NEXT, 143, 598, 592, 742, 1139
 CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:PEAK, 144, 599, 592, 742, 1140
 CALCulate<Window>:DELTamarker<DeltaMarker>:MINimum:RIGHT, 599, 593, 743, 1140

CALCulate<Window>:DELTaMarker<DeltaMarker>:MODEL, 1141
 CALCulate<Window>:DELTaMarker<DeltaMarker>:MREF, 1142
 CALCulate<Window>:DELTaMarker<DeltaMarker>:MREF:ENable, 145, 1143
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:FRAmE, 593, 1144
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:SAReDo, 595, 1145
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD, 596, 1146
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK, 597, 1147
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:ABOVE, 598, 1148
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue, 599, 1149
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:AUtO, 599, 1149
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:IMMediate, 599, 1149
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:MARGIn, 600, 1150
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:PSHov, 601, 1151
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:PSHov:FLow, 602, 1151
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:PSHov:FLow:FLINE<FreqLine>, 602, 1152
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:PSHov:FLow:FLINE<FreqLine>:STATE, 602, 1152
 CALCulate<Window>:DELTaMarker<DeltaMarker>:SPECtRum:WIdThInD:SPeAK:ELIN:VALue:PSearch:PSHov:FLow:FLINE<FreqLine>:FORMAt, 603, 1152
 CALCulate<Window>:DELTaMarker<DeltaMarker>:STATE, 146, 401, 603, 743, 1153
 CALCulate<Window>:DELTaMarker<DeltaMarker>:TRACe, 147, 402, 604, 744, 1154
 CALCulate<Window>:DELTaMarker<DeltaMarker>:X, 148, 403, 605, 1155
 CALCulate<Window>:DELTaMarker<DeltaMarker>:X:ABSOLute, 745
 CALCulate<Window>:DELTaMarker<DeltaMarker>:X:REALtIme, 404, 606, 746, 1156
 CALCulate<Window>:DELTaMarker<DeltaMarker>:Y, 149, 364, 404, 607, 746, 1156
 CALCulate<Window>:DISTRibution:NBINs, 608
 CALCulate<Window>:DLABs:STATE, 747
 CALCulate<Window>:DLABs:VALue, 748
 CALCulate<Window>:DLINe<DisplayLine>, 405
 CALCulate<Window>:DLINe<DisplayLine>:STATE, 406
 CALCulate<Window>:DLRel:STATE, 749
 CALCulate<Window>:DLRel:VALue, 750
 CALCulate<Window>:DSP:RESult:CAPTure:BURSts, 751
 CALCulate<Window>:DSP:RESult:CAPTure:PATterns, 752
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:LENGth, 753
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:PRESEnt, 754
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:BURSt:STARt, 754
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:CONFId, 755
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:CORRe, 756
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:PRESEr, 756
 CALCulate<Window>:DSP:RESult:RRANge:CURRent:PATtern:STARt, 757
 CALCulate<Window>:ELIN:VALue, 759
 CALCulate<Window>:FSK:DEVIation:REFerence:RELative, 763
 CALCulate<Window>:FSK:DEVIation:REFerence:VALue, 764
 CALCulate<Window>:HOPDetection:PNOise:FREQuency:STARt, 609
 CALCulate<Window>:HOPDetection:PNOise:FREQuency:STOP, 610
 CALCulate<Window>:HOPDetection:PNOise:LENGth, 611
 CALCulate<Window>:HOPDetection:PNOise:OFFSet:BEGIN, 612
 CALCulate<Window>:HOPDetection:PNOise:OFFSet:END, 613
 CALCulate<Window>:HOPDetection:PNOise:REFerence, 613
 CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:RESult, 766
 CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:STATE, 767
 CALCulate<Window>:LIMit:MACCuracy:CFERror:CURRent:VALue, 767
 CALCulate<Window>:LIMit:MACCuracy:CFERror:MEAN:RESult, 768

CALCulate<Window>:LIMit:MACCuracy:CFError:MEAN:STATE, CALCulate<Window>:LIMit:MACCuracy:FDError:MEAN:RESult,
 769 790
 CALCulate<Window>:LIMit:MACCuracy:CFError:MEAN:VALUE, CALCulate<Window>:LIMit:MACCuracy:FDError:MEAN:STATE,
 770 791
 CALCulate<Window>:LIMit:MACCuracy:CFError:PEAK:RESult, CALCulate<Window>:LIMit:MACCuracy:FDError:MEAN:VALUE,
 771 792
 CALCulate<Window>:LIMit:MACCuracy:CFError:PEAK:STATE, CALCulate<Window>:LIMit:MACCuracy:FDError:PEAK:RESult,
 771 793
 CALCulate<Window>:LIMit:MACCuracy:CFError:PEAK:VALUE, CALCulate<Window>:LIMit:MACCuracy:FDError:PEAK:STATE,
 772 793
 CALCulate<Window>:LIMit:MACCuracy:DEFault, CALCulate<Window>:LIMit:MACCuracy:FDError:PEAK:VALUE,
 773 794
 CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:PCURrent:RESult,
 774 795
 CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:PCURrent:STATE,
 774 796
 CALCulate<Window>:LIMit:MACCuracy:EVM:PCURrent:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:PCURrent:VALUE,
 775 797
 CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:PMEan:RESult,
 776 798
 CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:PMEan:STATE,
 777 798
 CALCulate<Window>:LIMit:MACCuracy:EVM:PMEan:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:PMEan:VALUE,
 777 799
 CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:PPEak:RESult,
 778 800
 CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:PPEak:STATE,
 779 801
 CALCulate<Window>:LIMit:MACCuracy:EVM:PPEak:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:PPEak:VALUE,
 780 801
 CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:RCURrent:RESult,
 781 802
 CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:RCURrent:STATE,
 781 803
 CALCulate<Window>:LIMit:MACCuracy:EVM:RCURrent:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:RCURrent:VALUE,
 782 804
 CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:RMEan:RESult,
 783 805
 CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:RMEan:STATE,
 784 805
 CALCulate<Window>:LIMit:MACCuracy:EVM:RMEan:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:RMEan:VALUE,
 784 806
 CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:RESult, CALCulate<Window>:LIMit:MACCuracy:FError:RPEak:RESult,
 785 807
 CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:STATE, CALCulate<Window>:LIMit:MACCuracy:FError:RPEak:STATE,
 786 808
 CALCulate<Window>:LIMit:MACCuracy:EVM:RPEak:VALUE, CALCulate<Window>:LIMit:MACCuracy:FError:RPEak:VALUE,
 787 808
 CALCulate<Window>:LIMit:MACCuracy:FDError:CURREnt:RESult, CALCulate<Window>:LIMit:MACCuracy:MError:PCURrent:RESult,
 788 810
 CALCulate<Window>:LIMit:MACCuracy:FDError:CURREnt:STATE, CALCulate<Window>:LIMit:MACCuracy:MError:PCURrent:STATE,
 789 810
 CALCulate<Window>:LIMit:MACCuracy:FDError:CURREnt:VALUE, CALCulate<Window>:LIMit:MACCuracy:MError:PCURrent:VALUE,
 789 811

CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:RESult, 812
 CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:STATE, 813
 CALCulate<Window>:LIMit:MACCuracy:MERRor:PMEan:VALue, 813
 CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:RESult, 814
 CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:STATE, 815
 CALCulate<Window>:LIMit:MACCuracy:MERRor:PPEak:VALue, 816
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:RESult, 817
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:STATE, 817
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RCURrent:VALue, 818
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:RESult, 819
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:STATE, 820
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RMEan:VALue, 820
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:RESult, 821
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:STATE, 822
 CALCulate<Window>:LIMit:MACCuracy:MERRor:RPEak:VALue, 823
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:CURrent:RESult, 824
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:CURrent:STATE, 825
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:CURrent:VALue, 825
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:MEAN:RESult, 826
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:MEAN:STATE, 827
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:MEAN:VALue, 828
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:PEAK:RESult, 829
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:PEAK:STATE, 829
 CALCulate<Window>:LIMit:MACCuracy:OOFfset:PEAK:VALue, 830
 CALCulate<Window>:LIMit:MACCuracy:PERror:PCURrent:RESult, 831
 CALCulate<Window>:LIMit:MACCuracy:PERror:PCURrent:STATE, 832
 CALCulate<Window>:LIMit:MACCuracy:PERror:PCURrent:VALue, 833

1164	CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel<Window>:STATE, LIMit<LimitIx>:ACPower:GAP<GapChannel>:M
1165	CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel<Window>:STATE, LIMit<LimitIx>:ACPower:GAP<GapChannel>:M
1166	CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel<Window>:STATE, LIMit<LimitIx>:ACPower:GAP<GapChannel>:M
1167	CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel<Window>:STATE, LIMit<LimitIx>:ACPower:GAP<GapChannel>:M
1167	CALCulate<Window>:LIMit<LimitIx>:ACPower:ACHannel<Window>:STATE, LIMit<LimitIx>:ACPower:GAP<GapChannel>:M
1168	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:ACPower:GAP<GapChannel>:M
1170	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:ACPower:GAP<GapChannel>:M
1171	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:ACPower:STATE,
1172	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:ACTIVE, 150,
1174	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:CLEAR:IMMediate,
1174	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:COMMENT,
1175	CALCulate<Window>:LIMit<LimitIx>:ACPower:ALTErChannelIndoW:RESuLt<LimitIx>:CONTROL:DATA,
1177	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:CONTROL:DOMAIN,
1178	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:CONTROL:MODE,
1179	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:CONTROL:OFFSET,
1180	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:CONTROL:SHIFT,
1182	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:CONTROL:SPACING,
1183	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:COPY, 154,
1184	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:DELETE, 150,
1186	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:LIMIT
1187	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:MODE,
1188	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:PCLAS
1189	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:PCLAS
1190	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:PCLAS
1192	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:PCLAS
1193	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:PCLAS
1195	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:RESTO
	CALCulate<Window>:LIMit<LimitIx>:ACPower:GAP<GapChannel>:WINDOW:RESuLt<LimitIx>:ESPECTRUM<SubBlock>:VALUE

1226	CALCulate<Window>:LIMit<LimitIx>:FAIL, 155,	CALCulate<Window>:MARKer<Marker>:COUNT:FREQuency,
410, 1227		1250
CALCulate<Window>:LIMit<LimitIx>:LOWer:DATA,		CALCulate<Window>:MARKer<Marker>:COUNT:RESolution,
156, 1227		1250
CALCulate<Window>:LIMit<LimitIx>:LOWer:MARGin,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:AFRequenc
1228		1252
CALCulate<Window>:LIMit<LimitIx>:LOWer:MODE,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:AM:RESult
1229		1253
CALCulate<Window>:LIMit<LimitIx>:LOWer:OFFSet,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:AM:RESult
1230		1254
CALCulate<Window>:LIMit<LimitIx>:LOWer:SHIFt,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:CARRier:F
157, 1231		1255
CALCulate<Window>:LIMit<LimitIx>:LOWer:SPACing,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:DISTortio
1232		1257
CALCulate<Window>:LIMit<LimitIx>:LOWer:STATe,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:FERRor:RE
158, 1233		1260
CALCulate<Window>:LIMit<LimitIx>:LOWer:THReshold,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:FM:RESult
1234		1258
CALCulate<Window>:LIMit<LimitIx>:NAME, 159,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:FM:RESult
1235		1259
CALCulate<Window>:LIMit<LimitIx>:STATe, 160,		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:PM:RESult
1235		1261
CALCulate<Window>:LIMit<LimitIx>:TRACe, 161		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:PM:RESult
CALCulate<Window>:LIMit<LimitIx>:TRACe<Trace>, 1237		1262
CALCulate<Window>:LIMit<LimitIx>:TRACe<Trace>:CHECK, 1238		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:SINad:RES
		1264
CALCulate<Window>:LIMit<LimitIx>:TYPE, 162		CALCulate<Window>:MARKer<Marker>:FUNCTion:ADEMod:THD:RESul
CALCulate<Window>:LIMit<LimitIx>:UNIT, 1239		1265
CALCulate<Window>:LIMit<LimitIx>:UPPer:DATA,		CALCulate<Window>:MARKer<Marker>:FUNCTion:AFPHase:RESult,
163, 1240		1266
CALCulate<Window>:LIMit<LimitIx>:UPPer:MARGin, 1241		CALCulate<Window>:MARKer<Marker>:FUNCTion:AFPHase:STATe,
		1267
CALCulate<Window>:LIMit<LimitIx>:UPPer:MODE, 1242		CALCulate<Window>:MARKer<Marker>:FUNCTion:BPOWER:AOff,
		1268
CALCulate<Window>:LIMit<LimitIx>:UPPer:OFFSet, 1243		CALCulate<Window>:MARKer<Marker>:FUNCTion:BPOWER:MODE,
		1268
CALCulate<Window>:LIMit<LimitIx>:UPPer:SHIFt, 164, 1244		CALCulate<Window>:MARKer<Marker>:FUNCTion:BPOWER:RESult,
		1269
CALCulate<Window>:LIMit<LimitIx>:UPPer:SPACing, 1245		CALCulate<Window>:MARKer<Marker>:FUNCTion:BPOWER:SPAN,
		1270
CALCulate<Window>:LIMit<LimitIx>:UPPer:STATe, 164, 1246		CALCulate<Window>:MARKer<Marker>:FUNCTion:BPOWER:STATe,
		1271
CALCulate<Window>:LIMit<LimitIx>:UPPer:THReshold, 1247		CALCulate<Window>:MARKer<Marker>:FUNCTion:CENTer,
		1272
CALCulate<Window>:MARKer:AOff, 1248		CALCulate<Window>:MARKer<Marker>:FUNCTion:CSTep,
CALCulate<Window>:MARKer:LINK, 412		1272
CALCulate<Window>:MARKer:PEXCursion, 419, 622		CALCulate<Window>:MARKer<Marker>:FUNCTion:DDEMod:RESult,
CALCulate<Window>:MARKer:SPEctrogram:SARea,		855
624		CALCulate<Window>:MARKer<Marker>:FUNCTion:DDEMod:STATistic
CALCulate<Window>:MARKer<Marker>:AOff, 166,		856
411, 615, 854		CALCulate<Window>:MARKer<Marker>:FUNCTion:DDEMod:STATistic
CALCulate<Window>:MARKer<Marker>:COUNT, 1249		857
		CALCulate<Window>:MARKer<Marker>:FUNCTion:DDEMod:STATistic
		857

CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:HARMonics:NHARM	858	1283
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:HARMonics:PRESet	859	1280
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:HARMonics:STATE,	860	1284
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:MDEPth:RESult<Tr	860	1285
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:MDEPth:STATE,	861	1286
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:MSUMmary,	862	1287
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown,	862	1288
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown:FREQuenc	863	1289
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown:QFACTOR,	864	1289
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown:RESult,	864	1290
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown:STATE,	865	1290
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NDBDown:TIME,	866	1291
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NOISE:AOff,	866	1292
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NOISE:RESult,	867	1293
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:NOISE:STATE,	868	1293
CALCulate<Window>:MARKer<Marker>:FUNCTION:DDEMOAL:STATis<Window>:MARKer<Marker>:FUNCTION:PNOise:AOff,	868	1294
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:PNOise:RESult,	1273	1295
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:PNOise:STATE,	1274	1295
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1275	1297
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1276	1298
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1277	1296
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1278	1299
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1279	1300
CALCulate<Window>:MARKer<Marker>:FUNCTION:FPEAKs:CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1279	1301
CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:BAW<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1281	1302
CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:DIS<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1281	1303
CALCulate<Window>:MARKer<Marker>:FUNCTION:HARMonics:STATis<Window>:MARKer<Marker>:FUNCTION:POWER<SubBlock>:	1282	1304

CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER:CALCulate<Window>:MARKer<Marker>:FUNCTION:TOI:RESult,
 1304 1328
 CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER:CALCulate<Window>:MARKer<Marker>:FUNCTION:TOI:RESult:MAXim
 1305 1329
 CALCulate<Window>:MARKer<Marker>:FUNCTION:POWER:CALCulate<Window>:MARKer<Marker>:FUNCTION:TOI:RESult:MINim
 1306 1329
 CALCulate<Window>:MARKer<Marker>:FUNCTION:REFERENCE:CALCulate<Window>:MARKer<Marker>:FUNCTION:TOI:STATE,
 1307 1330
 CALCulate<Window>:MARKer<Marker>:FUNCTION:STRAck:STATE:CALCulate<Window>:MARKer<Marker>:LINK, 869,
 1308 1331
 CALCulate<Window>:MARKer<Marker>:FUNCTION:STRAck:UPDATE:CALCulate<Window>:MARKer<Marker>:LINK:TO:MARKer<Marker>Dest
 1309 413, 616, 1332
 CALCulate<Window>:MARKer<Marker>:FUNCTION:STRAck:UPDATE:CALCulate<Window>:MARKer<Marker>:LOEXclude,
 1310 1333
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:DIFF:CALCulate<Window>:MARKer<Marker>:MAXimum:APEak,
 1311 870
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage:CALCulate<Window>:MARKer<Marker>:MAXimum:AUTO,
 1311 1334
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MAXimum:LEFT,
 1313 414, 618, 871, 1334
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MAXimum:NEXT,
 1314 167, 415, 618, 871, 1335
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MAXimum:PEAK,
 1314 167, 416, 619, 872, 1336
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MAXimum:RIGHT,
 1315 416, 619, 872, 1336
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PHOLd:CALCulate<Window>:MARKer<Marker>:MBURst:START,
 1316 873
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PEAK:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MINimum:AUTO,
 1317 1337
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PEAK:PHOLd:RESult:CALCulate<Window>:MARKer<Marker>:MINimum:LEFT,
 1318 417, 620, 874, 1338
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PEAK:RESult:CALCulate<Window>:MARKer<Marker>:MINimum:NEXT,
 1319 168, 418, 621, 874, 1338
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:PEAK:STATE:CALCulate<Window>:MARKer<Marker>:MINimum:PEAK,
 1319 169, 418, 621, 875, 1339
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:RMS:AVERage:RESult:CALCulate<Window>:MARKer<Marker>:MINimum:RIGHT,
 1321 419, 622, 875, 1339
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:RMS:PHOLd:RESult:CALCulate<Window>:MARKer<Marker>:PEXCursion,
 1321 1340
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:RMS:RESult:CALCulate<Window>:MARKer<Marker>:SEARCH, 876
 1322 CALCulate<Window>:MARKer<Marker>:SPECTrogram:FRAME,
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:RMS:STATE:41
 1323 CALCulate<Window>:MARKer<Marker>:SPECTrogram:SAREa,
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:AVERage:RESult,
 1324 CALCulate<Window>:MARKer<Marker>:SPECTrogram:XY:MAXimum:PE
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:PHOLd:RESult,
 1325 CALCulate<Window>:MARKer<Marker>:SPECTrogram:XY:MINimum:PE
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:RESult,
 1326 CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:ABC
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:SDEVIation:STATE,
 1326 CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:BEL
 CALCulate<Window>:MARKer<Marker>:FUNCTION:SUMMARY:STATE:8, 1346
 1327 CALCulate<Window>:MARKer<Marker>:SPECTrogram:Y:MAXimum:NEX

Index 2303

CALCulate<Window>:TLRel:STATE, 894
 CALCulate<Window>:TLRel:VALue, 895
 CALCulate<Window>:TRACe<Trace>:ADJust:ALIGnment:DEFaulT, 897
 CALCulate<Window>:TRACe<Trace>:ADJust:ALIGnment:OFFSet, 898
 CALCulate<Window>:TRACe<Trace>:ADJust:VALue, 899
 CALCulate<Window>:TRACe<Trace>:SYMBols, 900
 CALCulate<Window>:TRACe<Trace>:VALue, 901
 CALCulate<Window>:UNCertainty:COMMON, 173
 CALCulate<Window>:UNCertainty:DATA:FREQUency, 174
 CALCulate<Window>:UNCertainty:DATA:GAIN, 175
 CALCulate<Window>:UNCertainty:DATA:NOISe, 176
 CALCulate<Window>:UNCertainty:DATA:RESults, 176
 CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty, 178
 CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty:MODE, 179
 CALCulate<Window>:UNCertainty:ENR:CALibration:UNCertainty:POWER, 180
 CALCulate<Window>:UNCertainty:ENR:UNCertainty, 180
 CALCulate<Window>:UNCertainty:ENR:UNCertainty:CLEAR, 181
 CALCulate<Window>:UNCertainty:ENR:UNCertainty:CONF, 182
 CALCulate<Window>:UNCertainty:MATCH:DUT:IN:RL, 184
 CALCulate<Window>:UNCertainty:MATCH:DUT:IN:VSWR, 185
 CALCulate<Window>:UNCertainty:MATCH:DUT:OUT:RL, 186
 CALCulate<Window>:UNCertainty:MATCH:DUT:OUT:VSWR, 186
 CALCulate<Window>:UNCertainty:MATCH:PREamp:RL, 187
 CALCulate<Window>:UNCertainty:MATCH:PREamp:VSWR, 188
 CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:RL, 189
 CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:SMSE, 190
 CALCulate<Window>:UNCertainty:MATCH:SOURce:CALibration:VSWR, 191
 CALCulate<Window>:UNCertainty:MATCH:SOURce:RL, 192
 CALCulate<Window>:UNCertainty:MATCH:SOURce:SNS, 192
 CALCulate<Window>:UNCertainty:MATCH:SOURce:VSWR, 193
 CALCulate<Window>:UNCertainty:PREamp:GAIN, 194
 CALCulate<Window>:UNCertainty:PREamp:NOISe, 195
 CALCulate<Window>:UNCertainty:PREamp:STATE, 196
 CALCulate<Window>:UNCertainty:RESult, 196
 CALCulate<Window>:UNCertainty:SANalyzer:GAIN:UNCertainty, 197
 CALCulate<Window>:UNCertainty:SANalyzer:NOISe:UNCertainty, 198
 CALCulate<Window>:UNIT:ANGLE, 902, 1394
 CALCulate<Window>:UNIT:POWER, 903, 1395
 CALCulate<Window>:X:UNIT:TIME, 904
 CALCulate<Window>:Y:UNIT:TIME, 905
 CALibration:AIQ:HATiming:STATE, 1396
 CALibration:ALL, 1397
 CALibration:DUE:DAYS, 1398
 CALibration:DUE:SCHedule, 1399
 CALibration:DUE:SHUTdown, 1400
 CALibration:MODE, 1400
 CALibration:POWER:PowerMeter<PowerMeter>:ZERO:AUTO, 430
 CALibration:PRESelection, 1401
 CALibration:RESult, 1402
 clear_cached_entries() (*ScpiLogger method*), 2288
 clear_relative_timestamp() (*ScpiLogger method*), 2288
 CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:MODE, 1404
 CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:REFERENCE, 1405
 CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:REFERENCE:MEAS, 1406
 CONFigure:ADEMod:RESults:AM:DETEctor<Trace>:STATE, 1407
 CONFigure:ADEMod:RESults:FM:DETEctor<Trace>:MODE, 1408
 CONFigure:ADEMod:RESults:FM:DETEctor<Trace>:REFERENCE, 1409
 CONFigure:ADEMod:RESults:FM:DETEctor<Trace>:REFERENCE:MEAS, 1410
 CONFigure:ADEMod:RESults:FM:DETEctor<Trace>:STATE, 1411
 CONFigure:ADEMod:RESults:PM:DETEctor<Trace>:MODE, 1412
 CONFigure:ADEMod:RESults:PM:DETEctor<Trace>:REFERENCE, 1413
 CONFigure:ADEMod:RESults:PM:DETEctor<Trace>:REFERENCE:MEAS, 1414
 CONFigure:ADEMod:RESults:PM:DETEctor<Trace>:STATE, 1415
 CONFigure:ADEMod:RESults:UNIT, 1416
 CONFigure:GENERator:CONNEction:CState, 1417
 CONFigure:GENERator:CONNEction:STATE, 1418

CONFigure:GENerator:IPConnection:ADDRESS,
1418

CONFigure:GENerator:RECORDing:COMBine, 1419

D

default_mode (*ScpiLogger attribute*), 2287

device_name (*ScpiLogger attribute*), 2287

DIAGnostic:HUMS:ALL, 1421

DIAGnostic:HUMS:BIOS, 1421

DIAGnostic:HUMS:DELeTe:ALL, 1420

DIAGnostic:HUMS:DEvIce:HISTory, 1422

DIAGnostic:HUMS:DEvIce:HISTory:DELeTe:ALL,
1422

DIAGnostic:HUMS:EQUipment, 1423

DIAGnostic:HUMS:FORMat, 1423

DIAGnostic:HUMS:SAVE, 1420

DIAGnostic:HUMS:SECurity, 1424

DIAGnostic:HUMS:SERvice, 1424

DIAGnostic:HUMS:STATe, 1424

DIAGnostic:HUMS:STORAge, 1425

DIAGnostic:HUMS:SW, 1425

DIAGnostic:HUMS:SYSTem:INFO, 1426

DIAGnostic:HUMS:SYSTem:STATUS, 1426

DIAGnostic:HUMS:SYSTem:STATUS:SUMMARY, 1427

DIAGnostic:HUMS:TAgS:ALL, 1428

DIAGnostic:HUMS:TAgS:DELeTe, 1427

DIAGnostic:HUMS:TAgS:DELeTe:ALL, 1427

DIAGnostic:HUMS:TAgS:VALue, 1428

DIAGnostic:HUMS:UTILization, 1429

DIAGnostic:HUMS:UTILization:ACTivity, 1430

DIAGnostic:HUMS:UTILization:ACTivity:TRACking:STATe,
1431

DIAGnostic:HUMS:UTILization:HISTory, 1432

DIAGnostic:HUMS:UTILization:HISTory:DELeTe:ALL,
1432

DIAGnostic:INFO:CCOunt, 1433

DIAGnostic:SERvice:BIOSinfo, 1434

DIAGnostic:SERvice:CALibration:DATE, 1434

DIAGnostic:SERvice:CALibration:DUE:DATE, 1435

DIAGnostic:SERvice:CALibration:DUE:STATe,
1436

DIAGnostic:SERvice:CALibration:INTERval, 1436

DIAGnostic:SERvice:DATE, 1437

DIAGnostic:SERvice:HWInfo, 1437

DIAGnostic:SERvice:INPut:AIQ:TYPE, 1438

DIAGnostic:SERvice:INPut:EMI:CFRequency, 1439

DIAGnostic:SERvice:INPut:EMI:SPECTrum, 1440

DIAGnostic:SERvice:INPut:MC:CFRequency, 1441

DIAGnostic:SERvice:INPut:MC:DISTance, 1441

DIAGnostic:SERvice:INPut:MC:RANGe, 1442

DIAGnostic:SERvice:INPut:PULSed:CFRequency,
1443

DIAGnostic:SERvice:INPut:PULSed:MCFRequency,
1443

DIAGnostic:SERvice:INPut:PULSed:WBFRequency,
1444

DIAGnostic:SERvice:INPut:RF:CFRequency, 1445

DIAGnostic:SERvice:INPut:RF:CPATH, 1445

DIAGnostic:SERvice:INPut:RF:SPECTrum, 1446

DIAGnostic:SERvice:INPut:SELeT, 1447

DIAGnostic:SERvice:INPut:SYNThtwo:FRequency,
1447

DIAGnostic:SERvice:NSource, 1448

DIAGnostic:SERvice:SFUNction, 1449

DIAGnostic:SERvice:SFUNction:LASTresult, 1450

DIAGnostic:SERvice:SFUNction:RESults:DELeTe,
1450

DIAGnostic:SERvice:SFUNction:RESults:SAVE,
1451

DIAGnostic:SERvice:SINFo, 1451

DIAGnostic:SERvice:SPCHeck:EXECute, 1452

DIAGnostic:SERvice:SPCHeck:RESult, 1453

DIAGnostic:SERvice:STATe, 1453

DIAGnostic:SERvice:STEST:RESult, 1454

DIAGnostic:SERvice:VERSinfo, 1454

DISPlay:ANNotation:CBAR, 1455

DISPlay:ANNotation:FRequency, 1456

DISPlay:ATAB, 1456

DISPlay:BLIGHting, 1457

DISPlay:CMAP<Item>:DEFault<Colors>, 1458

DISPlay:CMAP<Item>:HSL, 1459

DISPlay:CMAP<Item>:PDEFined, 1460

DISPlay:FACcess:POSition, 1461

DISPlay:FORMat, 1461

DISPlay:ITERm:STATe, 1462

DISPlay:LOGO, 1463

DISPlay:MINFo:STATe, 1464

DISPlay:MTABLE, 431, 640

DISPlay:PRESelector:POSition, 1464

DISPlay:SBAR:STATe, 1465

DISPlay:SKEYs:STATe, 1466

DISPlay:TBAR:STATe, 1467

DISPlay:THEMe:CATalog, 1468

DISPlay:THEMe:SELeT, 1468

DISPlay:TOUCHscreen:STATe, 1469

DISPlay:WINDow<Window>:ITEM:LINE:VALue, 907

DISPlay:WINDow<Window>:MINFo:STATe, 200, 432,
1470

DISPlay:WINDow<Window>:MTABLE, 200, 433, 1471

DISPlay:WINDow<Window>:PMETER<PowerMeter>:STATe,
434

DISPlay:WINDow<Window>:PRATe:AUTO, 909

DISPlay:WINDow<Window>:PRATe:VALue, 909

DISPlay:WINDow<Window>:SIZE, 201, 435, 641, 910,
1472

DISPlay:WINDow<Window>:SPECTrogram:COLor:DEFault,
1473

DISPLAY:WINDOW<Window>:SPECTrogram:COLor:LOWer 1474
 DISPLAY:WINDOW<Window>:SPECTrogram:COLor:SHAPE 1474
 DISPLAY:WINDOW<Window>:SPECTrogram:COLor:STYLE 643, 1475
 DISPLAY:WINDOW<Window>:SPECTrogram:COLor:UPPer 1476
 DISPLAY:WINDOW<Window>:STATE, 911, 1476
 DISPLAY:WINDOW<Window>:STATistics:CCDF:GAUSS, 1478
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:MODE, 1479
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:MODE:HCONTinuous, 1481
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:MODE:SELECT, 1482
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:MODE:HCONTinuous, 1483
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:PRESet, 1484
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:SPACing, 1484
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE, 1485
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:AUTO, 1487
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:MODE, 1487
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:PDIVision, 1488
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:RLEVEL, 1489
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:RLEVEL:OFFSet, 1491
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:RPOSITION, 1492
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:RVALUE, 1493
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SCALE:PDIVision, 1494
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:Y:SPACing, 1494
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:X:SCALE:RPOSITION, 1495
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:X:SCALE:RVALUE, 1497
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:X:SCALE:START, 1499
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:X:SCALE:STOP, 1500
 DISPLAY:WINDOW<Window>:SUBWindow<SubWindow>:TRACE<Trace>:X:SCALE:VOFFset, 1500
 DISPLAY:WINDOW<Window>:TABLE:ITEM, 202
 DISPLAY:WINDOW<Window>:TIME, 1501
 DISPLAY:WINDOW<Window>:TIME:FORMAT, 1502
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:AUTO, 439
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:MAXimum, 440
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:MINimum, 440
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:PDIVision, 441
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:RLEVEL, 649
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:RLEVEL:OFFSet, 650
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:RPOSITION, 442, 651
 DISPLAY:WINDOW<Window>:TRACE:Y:SCALE:RVALUE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:LENGTH, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:MODE:HCONTinuous, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:MODE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:MODE:HCONTinuous, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:MODE:SELECT, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:MODE:HCONTinuous, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:PRESet, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SPACing, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SELECT, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SMOothing:APERture, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SMOothing:STATE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:STATE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SYMBOL, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SYMBOLS, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:SYMBOLS:OFFSet, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:UNCertainty, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:PDIVision, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:RPOSITION, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:RVALUE, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:START, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:STOP, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:X:SCALE:VOFFset, 442, 651
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE, 438, 648, 1509
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:AUTO, 212, 1510
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:AUTO:ALL, 922

DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:BOTTOM, 213
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:MAXIMUM, 1511
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:MINIMUM, 1512
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:MODE, 922, 1513
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:PDIVISION, 923, 1514
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:RLEVEL, 214, 1515
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:RLEVEL:TRACE, 215
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:RLEVEL:TRACE:WINDOW:DATA, 924, 1516
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:RPOSITION, 925, 1517
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:RVFORMAT, 926, 1518
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SCALE:TOP, 216
 DISPLAY:WINDOW<Window>:TRACE<Trace>:Y:SPACING, 927, 1519
 DISPLAY:WSELECT, 443, 652, 1520

E

error() (*ScpiLogger method*), 2288

F

FETCH:PMETER<PowerMeter>, 444, 1521
 FETCH:PNOISE<Trace>:IPN, 373
 FETCH:PNOISE<Trace>:MEASURED:FREQUENCY, 374
 FETCH:PNOISE<Trace>:MEASURED:LEVEL, 375
 FETCH:PNOISE<Trace>:RFM, 375
 FETCH:PNOISE<Trace>:RMS, 376
 FETCH:PNOISE<Trace>:RPM, 376
 FETCH:PNOISE<Trace>:SPURS, 377
 FETCH:PNOISE<Trace>:SPURS:DISCRETE, 377
 FETCH:PNOISE<Trace>:SPURS:RANDOM, 378
 FETCH:PNOISE<Trace>:SWEep:AVG, 378
 FETCH:PNOISE<Trace>:SWEep:FDRIFT, 379
 FETCH:PNOISE<Trace>:SWEep:LDRIFT, 379
 FETCH:PNOISE<Trace>:SWEep:MDRIFT, 380
 FETCH:PNOISE<Trace>:SWEep:SRATE, 381
 FETCH:PNOISE<Trace>:SWEep:START, 380
 FETCH:PNOISE<Trace>:SWEep:STOP, 381
 FETCH:PNOISE<Trace>:USER<UserRange>:IPN, 382
 FETCH:PNOISE<Trace>:USER<UserRange>:RFM, 383
 FETCH:PNOISE<Trace>:USER<UserRange>:RMS, 383
 FETCH:PNOISE<Trace>:USER<UserRange>:RPM, 384
 flush() (*ScpiLogger method*), 2288
 FORMAT REAL, 32

SENSE:CORRECTION:FRESPONSE:USER:IQ:DATA:FREQUENCY, 1806
 SENSE:CORRECTION:FRESPONSE:USER:IQ:DATA:MAGNITUDE, 1807
 SENSE:CORRECTION:FRESPONSE:USER:IQ:DATA:PHASE, 1807
 SENSE:CORRECTION:FRESPONSE:USER:SPECTRUM:DATA:FREQUENCY, 1817
 SENSE:CORRECTION:FRESPONSE:USER:SPECTRUM:DATA:MAGNITUDE, 1818
 SENSE:CORRECTION:FRESPONSE:USER:SPECTRUM:DATA:PHASE, 1818
 TRACE:IQ:DATA, 2251
 TRACE:IQ:DATA:MEMORY, 2251, 2252
 TRACE:WINDOW:DATA, 356, 390, 566, 718, 1105, 2248
 TRACE:WINDOW:DATA:MEMORY, 2249
 TRACE:WINDOW:DATA:X, 566, 719, 2250
 WINDOW:DATA, 1521
 FORMAT:DEXPORT:CSEPARATOR, 217
 FORMAT:DEXPORT:DSEPARATOR, 218, 445, 653, 928, 1522
 FORMAT:DEXPORT:FORMAT, 219
 FORMAT:DEXPORT:HEADER, 219, 446, 654, 929, 1523
 FORMAT:DEXPORT:MODE, 930
 FORMAT:DEXPORT:TRACES, 220, 447, 654, 1524
 FORMAT:DEXPORT:XDISTIB, 1524

G

get_logging_target() (*ScpiLogger method*), 2287
 get_relative_timestamp() (*ScpiLogger method*), 2288

H

HCPY:ABORT, 1525
 HCPY:CMAP<Item>:DEFAULT<Colors>, 1526
 HCPY:CMAP<Item>:HSL, 1527
 HCPY:CMAP<Item>:PDEFINED, 1528
 HCPY:COMMENT, 1529
 HCPY:CONTENT, 1529
 HCPY:COPIES, 1530
 HCPY:DESTINATION, 1531
 HCPY:DEVICE:COLOR, 1532
 HCPY:DEVICE:LANGUAGE, 1532
 HCPY:IMMEDIATE, 1533
 HCPY:IMMEDIATE:NEXT, 1534
 HCPY:ITEM:ALL, 1535
 HCPY:ITEM:WINDOW:TEXT, 1536
 HCPY:ITEM:WINDOW:TRACE:STATE, 1536
 HCPY:LOGO, 1537
 HCPY:MODE, 1538
 HCPY:PAGE:COUNT:STATE, 1539
 HCPY:PAGE:MARGIN:BOTTOM, 1540
 HCPY:PAGE:MARGIN:LEFT, 1540

HCOpy:PAGE:MARGin:RIGHT, 1541
 HCOpy:PAGE:MARGin:TOP, 1542
 HCOpy:PAGE:MARGin:UNIT, 1542
 HCOpy:PAGE:ORientation, 1543
 HCOpy:PAGE:WINDow:CHANnel:STATe, 1544
 HCOpy:PAGE:WINDow:COUNt, 1545
 HCOpy:PAGE:WINDow:SCALE, 1545
 HCOpy:PAGE:WINDow:STATe, 1546
 HCOpy:PRINT, 1547
 HCOpy:TDSTamp:STATe, 1548
 HCOpy:THEMe:SElect, 1549
 HCOpy:TREPort:APPend, 1550
 HCOpy:TREPort:DESCription, 1550
 HCOpy:TREPort:ITEM:DEFault, 1551
 HCOpy:TREPort:ITEM:HEADer:LINE<Line>:CONTRol, 1552
 HCOpy:TREPort:ITEM:HEADer:LINE<Line>:TEXT, 1553
 HCOpy:TREPort:ITEM:HEADer:LINE<Line>:TITLE, 1554
 HCOpy:TREPort:ITEM:HEADer:STATe, 1554
 HCOpy:TREPort:ITEM:LIST, 1555
 HCOpy:TREPort:ITEM:LOGO, 1555
 HCOpy:TREPort:ITEM:LOGO:CONTRol, 1556
 HCOpy:TREPort:ITEM:SElect, 1557
 HCOpy:TREPort:ITEM:TEMPlate:CATalog, 1558
 HCOpy:TREPort:ITEM:TEMPlate:DELeTe, 1557
 HCOpy:TREPort:ITEM:TEMPlate:LOAD, 1557
 HCOpy:TREPort:ITEM:TEMPlate:SAVE, 1557
 HCOpy:TREPort:NEW, 1558
 HCOpy:TREPort:PAGecount:STATe, 1559
 HCOpy:TREPort:PAGesize, 1560
 HCOpy:TREPort:PCOLors:STATe, 1560
 HCOpy:TREPort:TDSTamp:STATe, 1561
 HCOpy:TREPort:TEST:REMOve, 1562
 HCOpy:TREPort:TEST:REMOve:ALL, 1563
 HCOpy:TREPort:TEST:REMOve:SElected, 1563
 HCOpy:TREPort:TEST:SElect, 1564
 HCOpy:TREPort:TEST:SElect:ALL, 1564
 HCOpy:TREPort:TEST:SElect:INVert, 1565
 HCOpy:TREPort:TEST:SElect:NONE, 1565
 HCOpy:TREPort:TITLE, 1566
 HCOpy:TREPort:TITLE:STATe, 1567
 |
 info() (*ScpiLogger method*), 2288
 info_raw() (*ScpiLogger method*), 2288
 INITiate:BLOCK:ABORT, 1568
 INITiate:BLOCK:CONMeas, 1569
 INITiate:BLOCK:IMMediate, 1569
 INITiate:CONMeas, 448, 655, 930, 1570
 INITiate:CONTinuous, 221, 448, 656, 931, 1570
 INITiate:ESpectrum, 1571
 INITiate:IMMediate, 222, 449, 657, 932, 1567
 INITiate:REFMeas, 933
 INITiate:REFresh, 933
 INITiate:SEQuencer:ABORT, 1571
 INITiate:SEQuencer:IMMediate, 1572
 INITiate:SEQuencer:MODE, 1573
 INITiate:SEQuencer:REFresh:ALL, 1573
 INITiate:SPURious, 450, 1574
 INITiate:SYNC, 450
 INPut:ATTenuation, 658, 1575
 INPut:ATTenuation:AUTO, 659, 1576
 INPut:CONNector, 451, 659, 1577
 INPut:COUPling, 452, 660, 1577
 INPut:DIQ:CDEvice, 1578
 INPut:DIQ:RANGe:COUPling, 1579
 INPut:DIQ:RANGe:UPPer, 1579
 INPut:DIQ:RANGe:UPPer:AUTO, 1580
 INPut:DIQ:RANGe:UPPer:UNIT, 1580
 INPut:DIQ:SRATe, 1581
 INPut:DIQ:SRATe:AUTO, 1582
 INPut:DPATH, 660, 1582
 INPut:EATT, 1583
 INPut:EATT:AUTO, 1583
 INPut:EATT:STATe, 1584
 INPut:FILE:PATH, 1585
 INPut:FILTer:HPASs:STATe, 453, 663, 1587
 INPut:FILTer:YIG:STATe, 454, 664, 1588
 INPut:GAIN:STATe, 665, 1589
 INPut:GAIN:VALue, 666, 1590
 INPut:IMPedance, 454, 666, 1590
 INPut:IQ:BALanced:STATe, 1592
 INPut:IQ:FULLscale:AUTO, 1592
 INPut:IQ:FULLscale:LEVel, 1593
 INPut:IQ:TYPE, 1602
 INPut:SElect, 667
 INPut<InputIx>:ATTenuation, 223
 INPut<InputIx>:ATTenuation:AUTO:MODE, 935
 INPut<InputIx>:ATTenuation:PROTection:RESet, 1576
 INPut<InputIx>:CONNector, 224
 INPut<InputIx>:COUPling, 224, 936
 INPut<InputIx>:DPATH, 225, 936
 INPut<InputIx>:EGain:BYPass, 661
 INPut<InputIx>:EGain:STATe, 662, 1584
 INPut<InputIx>:FILE:ZPADing, 1586
 INPut<InputIx>:FILTer:HPASs:STATe, 226, 938
 INPut<InputIx>:FILTer:YIG:STATe, 227, 939
 INPut<InputIx>:GAIN:STATe, 228, 940
 INPut<InputIx>:GAIN:VALue, 229, 941
 INPut<InputIx>:IMPedance, 230
 INPut<InputIx>:IQ:BALanced:STATe, 100
 INPut<InputIx>:IQ:FULLscale:LEVel, 101
 INPut<InputIx>:IQ:IMPedance:PTYPE, 102
 INPut<InputIx>:IQ:OSC:BALanced:STATe, 103, 1594

INPut<InputIx>:IQ:OSC:CONState, 104
 INPut<InputIx>:IQ:OSC:COUpling, 104
 INPut<InputIx>:IQ:OSC:FULLscale:LEVel, 105, 1595
 INPut<InputIx>:IQ:OSC:IDN, 106
 INPut<InputIx>:IQ:OSC:SKEW:I, 107, 1596
 INPut<InputIx>:IQ:OSC:SKEW:I:INVerted, 108, 1597
 INPut<InputIx>:IQ:OSC:SKEW:Q, 109, 1598
 INPut<InputIx>:IQ:OSC:SKEW:Q:INVerted, 109, 1598
 INPut<InputIx>:IQ:OSC:SRATe, 1600
 INPut<InputIx>:IQ:OSC:STATe, 110, 1599
 INPut<InputIx>:IQ:OSC:TCPIp, 111, 1601
 INPut<InputIx>:IQ:OSC:TYPE, 111, 1601
 INPut<InputIx>:IQ:OSC:VDEvice, 112
 INPut<InputIx>:IQ:OSC:VFIRmware, 113
 INPut<InputIx>:IQ:TYPE, 113
 INPut<InputIx>:LOSCillator:SOURce, 1603
 INPut<InputIx>:LOSCillator:SOURce:EXtErnal:LEVel, 1604
 INPut<InputIx>:SANAlyzer:ATTenuation, 1605
 INPut<InputIx>:SANAlyzer:ATTenuation:AUTO, 1606
 INPut<InputIx>:SElect, 942, 1607
 INPut<InputIx>:TERMinator, 1607
 INPut<InputIx>:TYPE, 231
 INPut<InputIx>:UPOrt:STATe, 1608
 INPut<InputIx>:UPOrt:VALue, 1609
 INStrument:ABORt, 1609
 INStrument:COUPle:ABIMpedance, 1611
 INStrument:COUPle:ACDC, 1611
 INStrument:COUPle:ATTen, 1612
 INStrument:COUPle:AUNit, 1612
 INStrument:COUPle:BWIDth, 1613
 INStrument:COUPle:CENTer, 1613
 INStrument:COUPle:DEMod, 1614
 INStrument:COUPle:GAIN, 1614
 INStrument:COUPle:GENerator:CENTer:OFFSet, 1615
 INStrument:COUPle:GENerator:CENTer:STATe, 1616
 INStrument:COUPle:GENerator:RLEVel:OFFSet, 1617
 INStrument:COUPle:GENerator:RLEVel:STATe, 1617
 INStrument:COUPle:GENerator:STATe, 1618
 INStrument:COUPle:IMPedance, 1619
 INStrument:COUPle:LIMit, 1619
 INStrument:COUPle:LLINes, 1620
 INStrument:COUPle:MARKer, 1620
 INStrument:COUPle:PRESel, 1621
 INStrument:COUPle:RLEVel, 1621
 INStrument:COUPle:SPAN, 1622

INStrument:COUPle:VBW, 1622
 INStrument:CREate:DUPLicate, 1623
 INStrument:CREate:NEW, 1623
 INStrument:CREate:REPLace, 1624
 INStrument:DELeTe, 1609
 INStrument:LIST, 1624
 INStrument:MODE, 1625
 INStrument:NSElect, 1625
 INStrument:REName, 1626
 INStrument:SElect, 1626, 1627

L

LAYout:ADD:WINDow, 114, 232, 385, 456, 668, 729, 943, 1628
 LAYout:CATalog:WINDow, 115, 233, 386, 457, 669, 730, 944, 1629
 LAYout:DIRection, 1629
 LAYout:IDENtify:WINDow, 116, 234, 387, 457, 670, 731, 945, 1630
 LAYout:MOVE:WINDow, 235, 458, 946
 LAYout:REMOve:WINDow, 236, 459, 671, 947, 1631
 LAYout:REPLace:WINDow, 117, 236, 387, 460, 671, 732, 947, 1632
 LAYout:SPLitter, 237, 460, 672, 948, 1632
 log_status_check_ok (*ScpiLogger attribute*), 2288
 log_to_console (*ScpiLogger attribute*), 2287
 log_to_console_and_udp (*ScpiLogger attribute*), 2287
 log_to_udp (*ScpiLogger attribute*), 2287

M

MMEMemory:CATalog, 1634
 MMEMemory:CATalog:LONG, 1635
 MMEMemory:CDIRectory, 1637
 MMEMemory:CLEar:ALL, 1633
 MMEMemory:CLEar:STATe 1,, 1636
 MMEMemory:COMMENT, 1636
 MMEMemory:COPY, 1633
 MMEMemory:DELeTe:IMMediate, 1637
 MMEMemory:LOAD:AUTO 1,, 1639
 MMEMemory:LOAD:IQ:STATe 1,, 1639
 MMEMemory:LOAD:IQ:STReam, 673, 950, 1640
 MMEMemory:LOAD:IQ:STReam:AUTO, 674, 951, 1640
 MMEMemory:LOAD:IQ:STReam:LIST, 675, 951, 1641
 MMEMemory:LOAD:STATe 1,, 1641
 MMEMemory:LOAD:TYPE, 1642
 MMEMemory:LOAD<Window>:IQ:STATe, 950
 MMEMemory:LOAD<Window>:TFACtor, 1642
 MMEMemory:MDIRectory, 1643
 MMEMemory:MOVE, 1633
 MMEMemory:MSIS, 1643
 MMEMemory:NAME, 1644
 MMEMemory:NETWork:DISConnect, 1645
 MMEMemory:NETWork:MAP, 1646
 MMEMemory:NETWork:UNUSeddrives, 1647

MMEMory:NETWork:USEDdrives, 1647
 MMEMory:RAW, 1648
 MMEMory:RDIRECTory, 1638
 MMEMory:SElect:CHANnel:ITEM:ALL, 1649
 MMEMory:SElect:CHANnel:ITEM:DEfauLt, 1650
 MMEMory:SElect:CHANnel:ITEM:HWSettings, 1651
 MMEMory:SElect:CHANnel:ITEM:LINEs:ALL, 1652
 MMEMory:SElect:CHANnel:ITEM:NONE, 1653
 MMEMory:SElect:CHANnel:ITEM:SCData, 1653
 MMEMory:SElect:CHANnel:ITEM:SPECTrogram, 1654
 MMEMory:SElect:CHANnel:ITEM:TRACe:ACTive, 1655
 MMEMory:SElect:CHANnel:ITEM:TRANsdUcer:ALL, 1656
 MMEMory:SElect:CHANnel:ITEM:WEIGHting, 1656
 MMEMory:SElect:ITEM:ALL, 1657
 MMEMory:SElect:ITEM:CDATa, 1658
 MMEMory:SElect:ITEM:CSETup, 1659
 MMEMory:SElect:ITEM:DEfauLt, 1659
 MMEMory:SElect:ITEM:FINAl, 1660
 MMEMory:SElect:ITEM:HCOPy, 1660
 MMEMory:SElect:ITEM:HWSettings, 1661
 MMEMory:SElect:ITEM:LINEs:ACTive, 1662
 MMEMory:SElect:ITEM:LINEs:ALL, 1663
 MMEMory:SElect:ITEM:MACRos, 1663
 MMEMory:SElect:ITEM:NONE, 1664
 MMEMory:SElect:ITEM:SCData, 1664
 MMEMory:SElect:ITEM:SPECTrogram, 1665
 MMEMory:SElect:ITEM:TRACe:ACTive, 1666
 MMEMory:SElect:ITEM:TRANsdUcer:ACTive, 1667
 MMEMory:SElect:ITEM:TRANsdUcer:ALL, 1667
 MMEMory:SElect:ITEM:VIQData, 1668
 MMEMory:SElect:ITEM:WEIGHting, 1668
 MMEMory:STORE:IQ:STATe 1,, 1672
 MMEMory:STORE:SPUR:MEAS, 462
 MMEMory:STORE:STATe 1,, 1675
 MMEMory:STORE:STATe:NEXT, 1676
 MMEMory:STORE:TYPE, 1679
 MMEMory:STORE<Store>:IQ:COMMeNt, 952, 1670
 MMEMory:STORE<Store>:IQ:FORMat, 953, 1671
 MMEMory:STORE<Store>:IQ:RANGe, 1671
 MMEMory:STORE<Store>:IQ:STATe, 954
 MMEMory:STORE<Store>:IQ:TRACe, 954
 MMEMory:STORE<Store>:LIST, 1672
 MMEMory:STORE<Store>:PEAK, 1673
 MMEMory:STORE<Store>:SPECTrogram, 1674
 MMEMory:STORE<Store>:SPURious, 1674
 MMEMory:STORE<Store>:TABLe, 462, 1676
 MMEMory:STORE<Store>:TABLe:LIMit, 1677
 MMEMory:STORE<Store>:TFACtor, 1678
 MMEMory:STORE<Store>:TRACe, 239, 463, 675, 955
 MMEMory:STORE<Store>:TRACe 1,, 1679
 mode (*ScpiLogger* attribute), 2287

O

OUTPut:DIQ:STATe, 1684
 OUTPut:IQHS:CDEvice, 677
 OUTPut:TRIGger<TriggerPort>:DIRection, 464, 678, 1688
 OUTPut:TRIGger<TriggerPort>:LEVel, 465, 679, 1689
 OUTPut:TRIGger<TriggerPort>:OTYPE, 466, 680
 OUTPut:TRIGger<TriggerPort>:PULSe:IMMediate, 467, 681, 1691
 OUTPut:TRIGger<TriggerPort>:PULSe:LENGth, 467, 681, 1691
 OUTPut<OutputConnector>:ADEMod:ONLine:AF:CFrequency, 1681
 OUTPut<OutputConnector>:ADEMod:ONLine:SOURce, 1682
 OUTPut<OutputConnector>:ADEMod:ONLine:STATe, 1683
 OUTPut<OutputConnector>:IF:IFFrequency, 1685
 OUTPut<OutputConnector>:IF:SOURce, 956, 1686
 OUTPut<OutputConnector>:IQHS:SRATe, 677
 OUTPut<OutputConnector>:PROBe<Probe>:POWer, 1687
 OUTPut<OutputConnector>:TRIGger<TriggerPort>:DIRection, 240, 958
 OUTPut<OutputConnector>:TRIGger<TriggerPort>:LEVel, 241, 959
 OUTPut<OutputConnector>:TRIGger<TriggerPort>:OTYPE, 242, 960, 1690
 OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:IMMediate, 243, 961
 OUTPut<OutputConnector>:TRIGger<TriggerPort>:PULSe:LENGth, 244, 961
 OUTPut<OutputConnector>:UPORt:STATe, 1692
 OUTPut<OutputConnector>:UPORt:VALue, 1693

R

READ:PMETer<PowerMeter>, 469, 1695
 restore_format_string() (*ScpiLogger* method), 2288

S

ScpiLogger (class in *RsFswp.Internal.ScpiLogger*), 2287
 SENSE:ADEMod:ADCPrefilter, 1696
 SENSE:ADEMod:AF:CENTer, 1697
 SENSE:ADEMod:AF:COUPling, 1697
 SENSE:ADEMod:AF:SPAN, 1698
 SENSE:ADEMod:AF:SPAN:FULL, 1699
 SENSE:ADEMod:AF:START, 1699
 SENSE:ADEMod:AF:STOP, 1700
 SENSE:ADEMod:AM:ABSolute:AFSPepectrum:RESult, 1701

SENSE:ADEMod:AM:ABSolute:AFSPepectrum:TYPE, 1702
 SENSE:ADEMod:AM:ABSolute:TDOMain:RESult, 1703
 SENSE:ADEMod:AM:ABSolute:TDOMain:TYPE, 1704
 SENSE:ADEMod:AM:RELative:AFSPepectrum:RESult, 1705
 SENSE:ADEMod:AM:RELative:AFSPepectrum:TYPE, 1706
 SENSE:ADEMod:AM:RELative:TDOMain:RESult, 1707
 SENSE:ADEMod:AM:RELative:TDOMain:TYPE, 1708
 SENSE:ADEMod:FM:AFSPepectrum:RESult, 1709
 SENSE:ADEMod:FM:AFSPepectrum:TYPE, 1710
 SENSE:ADEMod:FM:OFFSet, 1710
 SENSE:ADEMod:FM:TDOMain:RESult, 1711
 SENSE:ADEMod:FM:TDOMain:TYPE, 1712
 SENSE:ADEMod:MTIME, 1713
 SENSE:ADEMod:PM:AFSPepectrum:RESult, 1714
 SENSE:ADEMod:PM:AFSPepectrum:TYPE, 1715
 SENSE:ADEMod:PM:RPOint:X, 1715
 SENSE:ADEMod:PM:RPOint:X:MODE, 1716
 SENSE:ADEMod:PM:TDOMain:RESult, 1717
 SENSE:ADEMod:PM:TDOMain:TYPE, 1718
 SENSE:ADEMod:PRESet:REStore, 1719
 SENSE:ADEMod:PRESet:STANdard, 1719
 SENSE:ADEMod:PRESet:STORe, 1720
 SENSE:ADEMod:SET, 1721
 SENSE:ADEMod:SPECTrum:BWIDth:RESolution, 1722
 SENSE:ADEMod:SPECTrum:RESult, 1723
 SENSE:ADEMod:SPECTrum:SPAN:MAXimum, 1724
 SENSE:ADEMod:SPECTrum:SPAN:ZOOM, 1725
 SENSE:ADEMod:SPECTrum:TYPE, 1725
 SENSE:ADEMod:SQUelch:LEVel, 1727
 SENSE:ADEMod:SQUelch:STATe, 1728
 SENSE:ADEMod:ZOOM:LENGth, 1728
 SENSE:ADEMod:ZOOM:LENGth:MODE, 1729
 SENSE:ADEMod:ZOOM:STARt, 1730
 SENSE:ADEMod:ZOOM:STATe, 1731
 SENSE:ADJust:ALL, 1732
 SENSE:ADJust:CARRier, 470
 SENSE:ADJust:CONFigure:DURation, 1733
 SENSE:ADJust:CONFigure:DURation:MODE, 1734
 SENSE:ADJust:CONFigure:FREQuency:LIMit:HIGH, 1735
 SENSE:ADJust:CONFigure:FREQuency:LIMit:LOW, 1735
 SENSE:ADJust:CONFigure:HYSTeresis:LOWer, 963, 1736
 SENSE:ADJust:CONFigure:HYSTeresis:UPPer, 964, 1737
 SENSE:ADJust:CONFigure:LEVel:DURation, 1738
 SENSE:ADJust:CONFigure:LEVel:DURation:MODE, 965, 1739
 SENSE:ADJust:CONFigure:LEVel:THReshold, 1740
 SENSE:ADJust:CONFigure:TRIGger, 1740
 SENSE:ADJust:FREQuency, 1741
 SENSE:ADJust:LEVel, 470, 682, 966, 1742
 SENSE:ADJust:SCALE:Y:AUTO:CONTInuous, 1743
 SENSE:AVERage:COUNt, 1744
 SENSE:AVERage:STATe<Status>, 1745
 SENSE:AVERage:TYPE, 1746
 SENSE:BWIDth:DEMod, 1747
 SENSE:BWIDth:DEMod:TYPE, 1747
 SENSE:BWIDth:LIST:DATA, 246
 SENSE:BWIDth:RESolution, 1748
 SENSE:BWIDth:RESolution:AUTO, 1749
 SENSE:BWIDth:RESolution:FFT, 1749
 SENSE:BWIDth:RESolution:RATio, 1750
 SENSE:BWIDth:RESolution:TYPE, 1750
 SENSE:BWIDth:VIDeo, 1751
 SENSE:BWIDth:VIDeo:AUTO, 1752
 SENSE:BWIDth:VIDeo:RATio, 1752
 SENSE:BWIDth:VIDeo:TYPE, 1753
 SENSE:CONFigure:CONTRol, 247
 SENSE:CONFigure:CORRection, 248
 SENSE:CONFigure:FREQuency:CONTInuous, 249
 SENSE:CONFigure:FREQuency:SINGLE, 250
 SENSE:CONFigure:LIST:CONTInuous, 250
 SENSE:CONFigure:LIST:SINGLE, 251
 SENSE:CONFigure:MEASurement, 252
 SENSE:CONFigure:MODE:DUT, 253
 SENSE:CONFigure:MODE:SYSTEM:IF:FREQuency, 254
 SENSE:CONFigure:MODE:SYSTEM:LO, 255
 SENSE:CONFigure:MODE:SYSTEM:LO:FREQuency, 255
 SENSE:CONFigure:SINGLE, 256
 SENSE:CORRection:COLlect:ACQuire, 1754
 SENSE:CORRection:CVL:BAND, 472, 684, 1755
 SENSE:CORRection:CVL:BIAS, 473, 685, 1756
 SENSE:CORRection:CVL:CATalog, 473, 685, 1757
 SENSE:CORRection:CVL:CLear, 471, 683, 1755
 SENSE:CORRection:CVL:COMMeNt, 474, 686, 1757
 SENSE:CORRection:CVL:DATA, 475, 686, 1758
 SENSE:CORRection:CVL:HARMonic, 475, 687, 1759
 SENSE:CORRection:CVL:MIXer, 476, 688, 1759
 SENSE:CORRection:CVL:PORTs, 477, 688, 1760
 SENSE:CORRection:CVL:SElect, 478, 689, 1761
 SENSE:CORRection:CVL:SNUMber, 478, 689, 1761
 SENSE:CORRection:ENR:CALibration:MODE, 258
 SENSE:CORRection:ENR:CALibration:SNS:SRNumber, 259
 SENSE:CORRection:ENR:CALibration:SPOT, 260
 SENSE:CORRection:ENR:CALibration:SPOT:COLD, 260
 SENSE:CORRection:ENR:CALibration:SPOT:HOT, 261
 SENSE:CORRection:ENR:CALibration:TABLE:SElect, 262
 SENSE:CORRection:ENR:CALibration:TYPE, 263
 SENSE:CORRection:ENR:COMMON, 264

SENSE:CORRection:ENR:MEASurement:MODE, 265 1772
 SENSE:CORRection:ENR:MEASurement:SNS:SRNumber, SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 266 1771
 SENSE:CORRection:ENR:MEASurement:SPOT, 266 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:SPOT:COLD, 1773
 267 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:SPOT:HOT, 1774
 268 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:TABLE:DATA, 1775
 269 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:TABLE:DELeTe, 1775
 268 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:TABLE:LIST, 1776
 270 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<TouchStone>
 SENSE:CORRection:ENR:MEASurement:TABLE:SELeCt, 1776
 270 SENSE:CORRection:FRESponse:BASEband:USER:STATe,
 SENSE:CORRection:ENR:MEASurement:TABLE:TEMPerature:DATA, 1777
 271 SENSE:CORRection:FRESponse:BASEband:USER:STORE,
 SENSE:CORRection:ENR:MEASurement:TABLE:TEMPerature:DELeTe, 1777
 271 SENSE:CORRection:FRESponse:INPut<InputIx>:USER:ADJust:RLEV
 SENSE:CORRection:ENR:MEASurement:TABLE:TEMPerature:LIST, 1779
 272 SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 SENSE:CORRection:ENR:MEASurement:TYPE, 273 1781
 SENSE:CORRection:ENR:SNS:AUTO:SRNumber, 274 SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 SENSE:CORRection:ENR:SNS:AUTO:STATe, 274 1780
 SENSE:CORRection:FRESponse:BASEband:USER:ADJust:RLEV, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1764 1782
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1765 1783
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1765 1784
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1766 1785
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1767 1785
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:FLISt<FileL
 1768 1786
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:LOAD,
 1769 1778
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:PRESet,
 1769 1778
 SENSE:CORRection:FRESponse:BASEband:USER:FLISt<FileL, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:REFresh,
 1770 1786
 SENSE:CORRection:FRESponse:BASEband:USER:LOAD, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch
 1762 1788
 SENSE:CORRection:FRESponse:BASEband:USER:PRESet, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch
 1762 1787
 SENSE:CORRection:FRESponse:BASEband:USER:REFresh, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch
 1770 1789
 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<Touch, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch
 1772 1787
 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<Touch, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch
 1771 1790
 SENSE:CORRection:FRESponse:BASEband:USER:SLISt<Touch, SENSE:CORRection:FRESponse:INPut<InputIx>:USER:SLISt<Touch

2313

SENSE:CORRection:TEMPerature, 290
 SENSE:CORRection:TEMPerature:CONTrol, 291
 SENSE:CORRection:TRANSDUCER:ACTive, 1822
 SENSE:CORRection:TRANSDUCER:ADJust:RLEVel:STATe, 1823
 SENSE:CORRection:TRANSDUCER:CATalog, 1823
 SENSE:CORRection:TRANSDUCER:COMMeNt, 1824
 SENSE:CORRection:TRANSDUCER:DATA, 1824
 SENSE:CORRection:TRANSDUCER:DELeTe, 1821
 SENSE:CORRection:TRANSDUCER:GEneRatE, 1825
 SENSE:CORRection:TRANSDUCER:INPut<InputIx>:ACTive, 1826
 SENSE:CORRection:TRANSDUCER:INPut<InputIx>:STATe, 1827
 SENSE:CORRection:TRANSDUCER:SCALing, 1827
 SENSE:CORRection:TRANSDUCER:SELeCt, 1828
 SENSE:CORRection:TRANSDUCER:STATe, 1828
 SENSE:CORRection:TRANSDUCER:UNIT, 1829
 SENSE:CREFeRence:FReFeRence, 479
 SENSE:CREFeRence:FReQuency, 480
 SENSE:CREFeRence:GUARd:INTeRval, 481
 SENSE:CREFeRence:GUARd:STATe, 482
 SENSE:CREFeRence:HARMonics:IDENtify, 482
 SENSE:CREFeRence:HARMonics:MNUmber, 483
 SENSE:CREFeRence:HARMonics:TOLerance, 484
 SENSE:CREFeRence:PDETeCt:RANGe:CeNteR, 485
 SENSE:CREFeRence:PDETeCt:RANGe:SPAN, 485
 SENSE:CREFeRence:PDETeCt:RANGe:STARt, 486
 SENSE:CREFeRence:PDETeCt:RANGe:STOP, 487
 SENSE:CREFeRence:PREFeRence, 487
 SENSE:CREFeRence:SRANGe, 488
 SENSE:CREFeRence:VALue, 488
 SENSE:DDEMod:APSK:NSTATe, 967
 SENSE:DDEMod:ASK:NSTATe, 968
 SENSE:DDEMod:BORDERing, 969
 SENSE:DDEMod:ECALc:MODE, 970
 SENSE:DDEMod:ECALc:OFFSeT, 970
 SENSE:DDEMod:EPRate:AUTO, 971
 SENSE:DDEMod:EPRate:VALue, 972
 SENSE:DDEMod:EQUAlizer:FILE:FORMat, 974
 SENSE:DDEMod:EQUAlizer:LENGth, 975
 SENSE:DDEMod:EQUAlizer:LOAD, 975
 SENSE:DDEMod:EQUAlizer:MODE, 976
 SENSE:DDEMod:EQUAlizer:RESeT, 973
 SENSE:DDEMod:EQUAlizer:SAVe, 977
 SENSE:DDEMod:EQUAlizer:STATe, 977
 SENSE:DDEMod:FACTory:VALue, 978
 SENSE:DDEMod:FILTeR:ALPHa, 979
 SENSE:DDEMod:FILTeR:REFeRence, 979
 SENSE:DDEMod:FILTeR:STATe, 980
 SENSE:DDEMod:FORMat, 981
 SENSE:DDEMod:FSK:NSTATe, 982
 SENSE:DDEMod:FSYnc:AUTO, 982
 SENSE:DDEMod:FSYnc:LEVel, 983
 SENSE:DDEMod:FSYnc:MODE, 984
 SENSE:DDEMod:FSYnc:RESeT, 985
 SENSE:DDEMod:KDATA:NAME, 985
 SENSE:DDEMod:KDATA:SER:LIMit, 986
 SENSE:DDEMod:KDATA:STATe, 987
 SENSE:DDEMod:MAPPING:CATalog, 987
 SENSE:DDEMod:MAPPING:VALue, 988
 SENSE:DDEMod:MFILTeR:ALPHa, 989
 SENSE:DDEMod:MFILTeR:AUTO, 989
 SENSE:DDEMod:MFILTeR:NAME, 990
 SENSE:DDEMod:MFILTeR:STATe, 991
 SENSE:DDEMod:MFILTeR:USER, 991
 SENSE:DDEMod:MSK:FORMat, 992
 SENSE:DDEMod:NORMALize:ADRoop, 993
 SENSE:DDEMod:NORMALize:CFDRift, 994
 SENSE:DDEMod:NORMALize:CHANnel, 994
 SENSE:DDEMod:NORMALize:FDERRor, 995
 SENSE:DDEMod:NORMALize:IQIMbalance, 995
 SENSE:DDEMod:NORMALize:IQOffseT, 996
 SENSE:DDEMod:NORMALize:SRError, 996
 SENSE:DDEMod:NORMALize:VALue, 997
 SENSE:DDEMod:OPTimization, 998
 SENSE:DDEMod:PATTeRn:APSK:NSTATe, 999
 SENSE:DDEMod:PATTeRn:ASK:NSTATe, 1000
 SENSE:DDEMod:PATTeRn:FORMat, 1001
 SENSE:DDEMod:PATTeRn:FRAME:EDIT, 1002
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:NEXT:BOOSting, 1004
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:NEXT:MODulation, 1004
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:PREVIOUS:BOOSting, 1005
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:PREVIOUS:MODulation, 1006
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:SAVe, 1002
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:STRUcture, 1007
 SENSE:DDEMod:PATTeRn:FRAME:EDIT:TEXT, 1009
 SENSE:DDEMod:PATTeRn:FRAME:LOAD, 1009
 SENSE:DDEMod:PATTeRn:FRAME:MODE, 1010
 SENSE:DDEMod:PATTeRn:MAPPING:CATalog, 1011
 SENSE:DDEMod:PATTeRn:MAPPING:VALue, 1012
 SENSE:DDEMod:PATTeRn:PSK:FORMat, 1013
 SENSE:DDEMod:PATTeRn:PSK:NSTATe, 1014
 SENSE:DDEMod:PATTeRn:QAM:FORMat, 1015
 SENSE:DDEMod:PATTeRn:QAM:NSTATe, 1016
 SENSE:DDEMod:PATTeRn:QPSK:FORMat, 1017
 SENSE:DDEMod:PATTeRn:STATe, 1018
 SENSE:DDEMod:PATTeRn:USER:NAME, 1019
 SENSE:DDEMod:PRATe, 1019
 SENSE:DDEMod:PRESeT:CALC, 1020
 SENSE:DDEMod:PRESeT:RLEVel, 1021
 SENSE:DDEMod:PRESeT:STANdard, 1022
 SENSE:DDEMod:PSK:FORMat, 1023

SENSE:DDEMod:PSK:NState, 1024
 SENSE:DDEMod:QAM:FORMat, 1025
 SENSE:DDEMod:QAM:NState, 1026
 SENSE:DDEMod:QPSK:FORMat, 1027
 SENSE:DDEMod:RLENgth:AUTO, 1028
 SENSE:DDEMod:RLENgth:SYMBOLs:VALue, 1029
 SENSE:DDEMod:RLENgth:VALue, 1030
 SENSE:DDEMod:SBAND, 1031
 SENSE:DDEMod:SEARCh:BURSt:AUTO, 1032
 SENSE:DDEMod:SEARCh:BURSt:CONFIgure:AUTO, 1033
 SENSE:DDEMod:SEARCh:BURSt:GLENgth:MINimum, 1034
 SENSE:DDEMod:SEARCh:BURSt:LENgth:MAXimum, 1035
 SENSE:DDEMod:SEARCh:BURSt:LENgth:MINimum, 1036
 SENSE:DDEMod:SEARCh:BURSt:MODE, 1036
 SENSE:DDEMod:SEARCh:BURSt:SKIP:FALLing, 1037
 SENSE:DDEMod:SEARCh:BURSt:SKIP:RISing, 1038
 SENSE:DDEMod:SEARCh:BURSt:STATe, 1038
 SENSE:DDEMod:SEARCh:BURSt:TOLerance, 1039
 SENSE:DDEMod:SEARCh:MBURst:CALC, 1040
 SENSE:DDEMod:SEARCh:MBURst:START:SAMPles, 1041
 SENSE:DDEMod:SEARCh:MBURst:START:SYMBOLs, 1041
 SENSE:DDEMod:SEARCh:PATtern:CONFIgure:AUTO, 1042
 SENSE:DDEMod:SEARCh:PATtern:SYNC:AUTO, 1043
 SENSE:DDEMod:SEARCh:PATtern:SYNC:STATe, 1044
 SENSE:DDEMod:SEARCh:PULSe:STATe, 1045
 SENSE:DDEMod:SEARCh:SYNC:AUTO, 1046
 SENSE:DDEMod:SEARCh:SYNC:CATalog, 1047
 SENSE:DDEMod:SEARCh:SYNC:COMMeNt, 1048
 SENSE:DDEMod:SEARCh:SYNC:COpy, 1045
 SENSE:DDEMod:SEARCh:SYNC:DATA, 1048
 SENSE:DDEMod:SEARCh:SYNC:DELeTe, 1045
 SENSE:DDEMod:SEARCh:SYNC:FOUND, 1049
 SENSE:DDEMod:SEARCh:SYNC:IQThreshold, 1049
 SENSE:DDEMod:SEARCh:SYNC:MODE, 1050
 SENSE:DDEMod:SEARCh:SYNC:NAME, 1051
 SENSE:DDEMod:SEARCh:SYNC:NState, 1051
 SENSE:DDEMod:SEARCh:SYNC:OFFSet, 1052
 SENSE:DDEMod:SEARCh:SYNC:PATtern, 1052
 SENSE:DDEMod:SEARCh:SYNC:PATtern:ADD, 1053
 SENSE:DDEMod:SEARCh:SYNC:PATtern:REMOve, 1054
 SENSE:DDEMod:SEARCh:SYNC:SELeCt, 1054
 SENSE:DDEMod:SEARCh:SYNC:STATe, 1055, 1830
 SENSE:DDEMod:SEARCh:SYNC:TEXT, 1055
 SENSE:DDEMod:SEARCh:TIME, 1056
 SENSE:DDEMod:SIGNAL:PATtern, 1057
 SENSE:DDEMod:SIGNAL:VALue, 1057
 SENSE:DDEMod:SRATE, 1062
 SENSE:DDEMod:STANDard:COMMeNt, 1059
 SENSE:DDEMod:STANDard:DELeTe, 1058
 SENSE:DDEMod:STANDard:PRESet:VALue, 1060
 SENSE:DDEMod:STANDard:SAVE, 1058
 SENSE:DDEMod:STANDard:SYNC:OFFSet:STATe, 1061
 SENSE:DDEMod:STANDard:SYNC:OFFSet:VALue, 1061
 SENSE:DDEMod:TFILter:ALPHA, 1063
 SENSE:DDEMod:TFILter:NAME, 1063
 SENSE:DDEMod:TFILter:STATe, 1064
 SENSE:DDEMod:TFILter:USER, 1065
 SENSE:DDEMod:TIME, 1065
 SENSE:DDEMod:TIME:AUTO, 1066
 SENSE:DDEMod:USER:NAME, 1067
 SENSE:DIRected:INPut:ATTenuation, 490
 SENSE:DIRected:INPut:GAIN:STATe, 491
 SENSE:DIRected:INPut:GAIN:VALue, 492
 SENSE:DIRected:LOAD, 489
 SENSE:DIRected:LOFFset, 492
 SENSE:DIRected:MFRBw, 493
 SENSE:DIRected:NFFT, 493
 SENSE:DIRected:PEXCursion, 494
 SENSE:DIRected:RLEVel, 494
 SENSE:DIRected:SAVE, 489
 SENSE:DIRected:SETTings, 495
 SENSE:ESpectrum<SubBlock>:BWID, 1831
 SENSE:ESpectrum<SubBlock>:FILTer:RRC:ALPHA, 1832
 SENSE:ESpectrum<SubBlock>:FILTer:RRC:STATe, 1833
 SENSE:ESpectrum<SubBlock>:HSPeed, 1834
 SENSE:ESpectrum<SubBlock>:MSR:APPLy, 1835
 SENSE:ESpectrum<SubBlock>:MSR:BAND, 1835
 SENSE:ESpectrum<SubBlock>:MSR:BCATegory, 1836
 SENSE:ESpectrum<SubBlock>:MSR:CLASs, 1837
 SENSE:ESpectrum<SubBlock>:MSR:GSM:CARRier, 1838
 SENSE:ESpectrum<SubBlock>:MSR:GSM:CPResent, 1839
 SENSE:ESpectrum<SubBlock>:MSR:LTE:CPResent, 1840
 SENSE:ESpectrum<SubBlock>:MSR:MPower, 1840
 SENSE:ESpectrum<SubBlock>:MSR:RFBWidth, 1841
 SENSE:ESpectrum<SubBlock>:PRESet:REStore, 1842
 SENSE:ESpectrum<SubBlock>:PRESet:STANDard, 1842
 SENSE:ESpectrum<SubBlock>:PRESet:STORe, 1843
 SENSE:ESpectrum<SubBlock>:RANGe<RangePy>:BANDwidth:RESolutiOn, 1845
 SENSE:ESpectrum<SubBlock>:RANGe<RangePy>:BANDwidth:VIDeo, 1846
 SENSE:ESpectrum<SubBlock>:RANGe<RangePy>:COUNt, 1847

SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:DELeTe, 1876
 1844 SENSe:FILTer<FilterPy>:CCIR:WEIGHted:STATe,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:FILTer:TYPE, 1877
 1848 SENSe:FILTer<FilterPy>:DEMPHasis:STATe, 1878
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:FREQuency:SHARPer<FilterPy>:DEMPHasis:TCONstant,
 1849 1879
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:FREQuency:SHARPer<FilterPy>:HPASS:FREQuency:ABSolute,
 1850 1880
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPUt:SENSe:FilterPy:HPASS:FREQuency:MANual,
 1851 1881
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPUt:SENSe:FilterPy:HPASS:STATe, 1882
 1852 SENSe:FILTer<FilterPy>:LPASS:FREQuency:ABSolute,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPUt:GAIN:STATe,
 1853 SENSe:FILTer<FilterPy>:LPASS:FREQuency:MANual,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INPUt:GAIN:VALUe,
 1854 SENSe:FILTer<FilterPy>:LPASS:FREQuency:RELative,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:INSert, 1885
 1855 SENSe:FILTer<FilterPy>:LPASS:STATe, 1886
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:HDL:ABSOLute:START:COMPonent>:ADD, 498
 1856 SENSe:FPLan:COMPonent<Component>:BCENTER, 498
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:HDL:ABSOLute:STOP:COMPonent>:BSPAN, 499
 1857 SENSe:FPLan:COMPonent<Component>:DELeTe, 497
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:HDL:RELAtive:START:COMPonent>:FACTOr, 500
 1859 SENSe:FPLan:COMPonent<Component>:IDENTity,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:RELAtive:START:ABS,
 1860 SENSe:FPLan:COMPonent<Component>:PORT<Port>:FREQuency,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:RELAtive:START:FUNCTION,
 1861 SENSe:FPLan:COMPonent<Component>:PORT<Port>:MHARmonic,
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:RELAtive:STOP,
 1862 SENSe:FPLan:COMPonent<Component>:TYPE, 504
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:RELAtive:STOP:ABS,
 1863 SENSe:FPLan:PREDicted:EXPort, 505
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:RELAtive:STOP:FUNCTION,
 1864 SENSe:FPLan:TRANsfer, 505
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:LIMit<LimitIndex>:FREQuency:ANNOtation, 1887
 1865 SENSe:FREQuency:CENTer, 292, 690, 1888
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:MLCALC:SENSe:FREQuency:CENTer:STEP, 691, 1888
 1866 SENSe:FREQuency:CENTer:STEP:AUTO, 1068, 1889
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:RLEVEL:SENSe:FREQuency:CENTer:STEP:LINK, 1889
 1867 SENSe:FREQuency:CENTer:STEP:LINK:FACTOr, 1890
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:SWEPT:SENSe:FREQuency:OFFSet, 506, 692, 1069, 1891
 1868 SENSe:FREQuency:POINts, 292
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:SWEPT:SENSe:FREQuency:SINGLE, 293
 1869 SENSe:FREQuency:SINGLE:COUPled, 294
 SENSe:ESpectrum<SubBlock>:RANGe<RangePy>:TRANsFER:SENSe:FREQuency:SPAN, 294, 1892
 1869 SENSe:FREQuency:SPAN:FULL, 1892
 SENSe:ESpectrum<SubBlock>:RRANGe, 1870 SENSe:FREQuency:START, 295, 1893
 SENSe:ESpectrum<SubBlock>:RTYPE, 1871 SENSe:FREQuency:STEP, 296
 SENSe:ESpectrum<SubBlock>:SCENTER, 1871 SENSe:FREQuency:STOP, 296, 1894
 SENSe:ESpectrum<SubBlock>:SCOUNT, 1872 SENSe:FREQuency:TABLE:DATA, 297
 SENSe:ESpectrum<SubBlock>:SSETUP, 1873 SENSe:IQ:BWIDth:MODE, 1895
 SENSe:FILTer<FilterPy>:AOFF, 1874 SENSe:IQ:BWIDth:RESolution, 1896
 SENSe:FILTer<FilterPy>:AWEIGHTed:STATe, 1875 SENSe:IQ:FFT:ALGorithm, 118
 SENSe:FILTer<FilterPy>:CCIR:UNWEIGHTed:STATe, SENSe:IQ:FFT:LENGth, 119

SENSE:IQ:FFT:WINDow:LENGth, 120
 SENSE:IQ:FFT:WINDow:OVERlap, 121
 SENSE:IQ:FFT:WINDow:TYPE, 121
 SENSE:IQ:IMPedance, 122
 SENSE:IQ:WBAND, 122
 SENSE:LIST:CLEar, 506
 SENSE:LIST:LOAD, 506
 SENSE:LIST:POWEr:RESult, 1897
 SENSE:LIST:POWEr:SEQUence, 1897
 SENSE:LIST:POWEr:SET, 1898
 SENSE:LIST:POWEr:STATE, 1899
 SENSE:LIST:RANGe<RangePy>:BANDwidth:AUTO, 509
 SENSE:LIST:RANGe<RangePy>:BANDwidth:RESolution, 510, 1901
 SENSE:LIST:RANGe<RangePy>:BANDwidth:VIDeo, 1902
 SENSE:LIST:RANGe<RangePy>:BREak, 1902
 SENSE:LIST:RANGe<RangePy>:COUNT, 510, 1903
 SENSE:LIST:RANGe<RangePy>:DELeTe, 508, 1900
 SENSE:LIST:RANGe<RangePy>:DETEctor, 1904
 SENSE:LIST:RANGe<RangePy>:FILTer:TYPE, 1905
 SENSE:LIST:RANGe<RangePy>:FREQuency:START, 511, 1906
 SENSE:LIST:RANGe<RangePy>:FREQuency:STOP, 512, 1907
 SENSE:LIST:RANGe<RangePy>:INPut:ATTenuation, 513, 1908
 SENSE:LIST:RANGe<RangePy>:INPut:ATTenuation:AUTO, 1909
 SENSE:LIST:RANGe<RangePy>:INPut:GAIN:STATE, 514, 1909
 SENSE:LIST:RANGe<RangePy>:INPut:GAIN:VALue, 515, 1910
 SENSE:LIST:RANGe<RangePy>:INSert, 516
 SENSE:LIST:RANGe<RangePy>:LIMit:START, 1911
 SENSE:LIST:RANGe<RangePy>:LIMit:STATE, 1912
 SENSE:LIST:RANGe<RangePy>:LIMit:STOP, 1913
 SENSE:LIST:RANGe<RangePy>:LOFFset, 516
 SENSE:LIST:RANGe<RangePy>:MFRBw, 517
 SENSE:LIST:RANGe<RangePy>:NFFT, 518
 SENSE:LIST:RANGe<RangePy>:PEXCursion, 518
 SENSE:LIST:RANGe<RangePy>:POINts:VALue, 1914
 SENSE:LIST:RANGe<RangePy>:RLEVel, 519, 1914
 SENSE:LIST:RANGe<RangePy>:SNRatio, 520
 SENSE:LIST:RANGe<RangePy>:SWEep:TIME, 1915
 SENSE:LIST:RANGe<RangePy>:SWEep:TIME:AUTO, 1916
 SENSE:LIST:RANGe<RangePy>:THREShold:START, 521
 SENSE:LIST:RANGe<RangePy>:THREShold:STOP, 522
 SENSE:LIST:RANGe<RangePy>:TRANsducer, 1917
 SENSE:LIST:RANGe<RangePy>:UARange, 522
 SENSE:LIST:SAVE, 506
 SENSE:LIST:XADJust, 1917
 SENSE:MEASure:POINts, 523
 SENSE:MIXer:BIAS:HIGH, 524, 693, 1918
 SENSE:MIXer:BIAS:LOW, 525, 694, 1919
 SENSE:MIXer:FREQuency:HANDover, 526, 694, 1920
 SENSE:MIXer:FREQuency:START, 526, 695, 1921
 SENSE:MIXer:FREQuency:STOP, 527, 695, 1921
 SENSE:MIXer:HARMonic:BAND, 528, 696, 1922
 SENSE:MIXer:HARMonic:BAND:PRESet, 528, 696, 1922
 SENSE:MIXer:HARMonic:HIGH, 529, 1923
 SENSE:MIXer:HARMonic:HIGH:STATE, 530, 697, 1924
 SENSE:MIXer:HARMonic:HIGH:VALue, 530, 698
 SENSE:MIXer:HARMonic:LOW, 531, 698, 1924
 SENSE:MIXer:HARMonic:TYPE, 531, 699, 1925
 SENSE:MIXer:IF, 1926
 SENSE:MIXer:LOPower, 532, 700, 1926
 SENSE:MIXer:LOSS:HIGH, 533, 700, 1927
 SENSE:MIXer:LOSS:LOW, 533, 701, 1927
 SENSE:MIXer:LOSS:TABLE:HIGH, 534, 702, 1928
 SENSE:MIXer:LOSS:TABLE:LOW, 535, 703, 1929
 SENSE:MIXer:PORTs, 536, 703, 1930
 SENSE:MIXer:RFOVerrange:STATE, 536, 704, 1930
 SENSE:MIXer:SIGNAL, 537, 705, 1931
 SENSE:MIXer:STATE, 538, 705, 1932
 SENSE:MIXer:THREShold, 538, 706, 1932
 SENSE:MPOWER:FTYPE, 1933
 SENSE:MPOWER:RESult:LIST, 1934
 SENSE:MPOWER:RESult:MIN, 1934
 SENSE:MPOWER:SEQUence, 1935
 SENSE:MSRA:CAPTure:OFFSet, 707, 1070, 1936
 SENSE:PMETER<PowerMeter>:DCYCLE:STATE, 539, 1938
 SENSE:PMETER<PowerMeter>:DCYCLE:VALue, 540, 1938
 SENSE:PMETER<PowerMeter>:FREQuency, 541, 1939
 SENSE:PMETER<PowerMeter>:FREQuency:LINK, 542, 1940
 SENSE:PMETER<PowerMeter>:MTIME, 543, 1941
 SENSE:PMETER<PowerMeter>:MTIME:AVERAge:COUNT, 544, 1942
 SENSE:PMETER<PowerMeter>:MTIME:AVERAge:STATE, 545, 1943
 SENSE:PMETER<PowerMeter>:ROFFset:STATE, 546, 1944
 SENSE:PMETER<PowerMeter>:SOFFset, 546, 1944
 SENSE:PMETER<PowerMeter>:STATE, 547, 1945
 SENSE:PMETER<PowerMeter>:TRIGger:DTIME, 1946
 SENSE:PMETER<PowerMeter>:TRIGger:HOLDoff, 1947
 SENSE:PMETER<PowerMeter>:TRIGger:HYSTEResis, 1947
 SENSE:PMETER<PowerMeter>:TRIGger:LEVel, 548, 1948
 SENSE:PMETER<PowerMeter>:TRIGger:SLOPe, 1949

SENSE:POWer:ACHannel:SPACing:UACHannel, 2008
 SENSE:POWer:ACHannel:SPACing:UALternate<UpperASChannel>, 2009
 SENSE:POWer:ACHannel:SSETup, 2009
 SENSE:POWer:ACHannel:TXChannel:COUNT, 2010
 SENSE:POWer:BWIDth, 2011
 SENSE:POWer:HSPeEd, 2011
 SENSE:POWer:NCORrection, 2012
 SENSE:POWer:TRACe, 2013
 SENSE:PROBe<Probe>:ID:PARTnumber, 298, 551, 708, 2014
 SENSE:PROBe<Probe>:ID:SRNumber, 299, 389, 552, 709, 2014
 SENSE:PROBe<Probe>:SETup:ATTRatio, 552, 709, 2015
 SENSE:PROBe<Probe>:SETup:CMOffset, 553, 710, 2016
 SENSE:PROBe<Probe>:SETup:DMOffset, 554, 711, 2017
 SENSE:PROBe<Probe>:SETup:MODE, 555, 712, 2017
 SENSE:PROBe<Probe>:SETup:NAME, 555, 712, 2018
 SENSE:PROBe<Probe>:SETup:NMOffset, 556, 713, 2018
 SENSE:PROBe<Probe>:SETup:PMODE, 557, 714, 2019
 SENSE:PROBe<Probe>:SETup:PMOffset, 557, 714, 2020
 SENSE:PROBe<Probe>:SETup:STATe, 558, 715, 2021
 SENSE:PROBe<Probe>:SETup:TYPE, 559, 716, 2021
 SENSE:RLENGth, 2022
 SENSE:ROSCillator:COUpling:BANDwidth, 2023
 SENSE:ROSCillator:COUpling:BANDwidth:MODE, 2024
 SENSE:ROSCillator:COUpling:MODE, 2024
 SENSE:ROSCillator:LBWidth, 2025
 SENSE:ROSCillator:O100, 2026
 SENSE:ROSCillator:O640, 2026
 SENSE:ROSCillator:OSYNc, 2027
 SENSE:ROSCillator:OUTPut<OutputConnector>, 2028
 SENSE:ROSCillator:PASSthrough, 2029
 SENSE:ROSCillator:SOURce, 2029
 SENSE:ROSCillator:SOURce:EAUTO, 2030
 SENSE:ROSCillator:TRANge, 2031
 SENSE:RTMS:CAPture:OFFSet, 2032
 SENSE:SAMPling:CLKio:OUTPut, 2033
 SENSE:SRATe, 2058
 SENSE:SSEarch:CONTRol, 559
 SENSE:SSEarch:FPLan, 560
 SENSE:SSEarch:FPLan:TOLerance, 561
 SENSE:SSEarch:MSPur, 561
 SENSE:SSEarch:RMARK, 562
 SENSE:SSEarch:RREMove, 563
 SENSE:SSEarch:STYPe, 563
 SENSE:SWAPiq, 123, 1071, 2034
 SENSE:SWEEP:COUNT, 300, 716, 2035
 SENSE:SWEEP:COUNT:CURRENT, 717, 1072, 2036
 SENSE:SWEEP:COUNT:VALue, 1073
 SENSE:SWEEP:DURation, 2036
 SENSE:SWEEP:EGATe, 300, 2037
 SENSE:SWEEP:EGATe:AUTO, 301, 2037
 SENSE:SWEEP:EGATe:CONTinuous:PCount, 302
 SENSE:SWEEP:EGATe:CONTinuous:PLENgtH, 303
 SENSE:SWEEP:EGATe:CONTinuous:STATe, 303
 SENSE:SWEEP:EGATe:HOLDoff, 304, 2038
 SENSE:SWEEP:EGATe:LENGth, 304, 2038
 SENSE:SWEEP:EGATe:LEVel:EXtERnal<ExternalPort>, 306, 2040
 SENSE:SWEEP:EGATe:LEVel:IFPower, 2040
 SENSE:SWEEP:EGATe:LEVel:RFPower, 2041
 SENSE:SWEEP:EGATe:POLarity, 306, 2042
 SENSE:SWEEP:EGATe:SOURce, 307, 2042
 SENSE:SWEEP:EGATe:TRACe<Trace>:COMMeNT, 2043
 SENSE:SWEEP:EGATe:TRACe<Trace>:PERiod, 2044
 SENSE:SWEEP:EGATe:TRACe<Trace>:START<GateRange>, 2045
 SENSE:SWEEP:EGATe:TRACe<Trace>:STATe<Status>, 2046
 SENSE:SWEEP:EGATe:TRACe<Trace>:STOP<GateRange>, 2047
 SENSE:SWEEP:EGATe:TYPE, 308, 2048
 SENSE:SWEEP:MODE, 2049
 SENSE:SWEEP:OPTimize, 2050
 SENSE:SWEEP:POINts, 2050
 SENSE:SWEEP:SCAPture:EVENTs, 2051
 SENSE:SWEEP:SCAPture:GAP, 2051
 SENSE:SWEEP:SCAPture:LENGth:TIME, 2052
 SENSE:SWEEP:SCAPture:OFFSet:TIME, 2053
 SENSE:SWEEP:SCAPture:STATe, 2054
 SENSE:SWEEP:TIME, 2054
 SENSE:SWEEP:TIME:AUTO, 309, 2055
 SENSE:SWEEP:TYPE, 2055
 SENSE:SWEEP:TYPE:USED, 2056
 SENSE:SWEEP:WINDow<Window>:POINts, 2057
 SENSE:TCAPture:LENGth, 1073
 SENSE:TRACE:IQ:SYNc:MODE, 2059
 SENSE:TRANsfer:SEGMeNT, 564
 SENSE:TRANsfer:SPUR, 565
 SENSE:WINDow<Window>:DETECTOR<Trace>:FUNCTION, 2060
 SENSE:WINDow<Window>:DETECTOR<Trace>:FUNCTION:AUTO, 2061
 set_format_string() (*ScpiLogger method*), 2288
 set_logging_target() (*ScpiLogger method*), 2287
 set_logging_target_global() (*ScpiLogger method*), 2287
 set_relative_timestamp() (*ScpiLogger method*), 2288

set_relative_timestamp_now() <i>method</i>), 2288	(ScpiLogger	SOURCE:VOLTage:AUX:LEVel:STATe, 330, 2091 SOURCE:VOLTage:CHANnel:COUPling, 331, 2092 SOURCE:VOLTage:CONTRol<Source>:LEVel:AMPLitude, 333, 2093 SOURCE:VOLTage:CONTRol<Source>:LEVel:LIMit:HIGH, 334, 2094 SOURCE:VOLTage:CONTRol<Source>:LEVel:LIMit:LOW, 335, 2095 SOURCE:VOLTage:CONTRol<Source>:LEVel:STATe, 335, 2096 SOURCE:VOLTage:POWer<Source>:LEVel:AMPLitude, 337, 2097 SOURCE:VOLTage:POWer<Source>:LEVel:LIMit:HIGH, 338, 2098 SOURCE:VOLTage:POWer<Source>:LEVel:LIMit:LOW, 339, 2099 SOURCE:VOLTage:POWer<Source>:LEVel:MODE, 339, 2100 SOURCE:VOLTage:POWer<Source>:LEVel:STATe, 340, 2101 SOURCE:VOLTage:POWer<Source>:LIMit:HIGH, 341, 2102 SOURCE:VOLTage:SEQUence:RESult, 342, 2103 SOURCE:VOLTage:STATe, 342, 2104 SOURCE:VOLTage:USEPort, 343 STATUS:OPERation:CONDition, 2106 STATUS:OPERation:ENABLE, 2106 STATUS:OPERation:EVENT, 2107 STATUS:OPERation:NTRansition, 2107 STATUS:OPERation:PCALibration:CONDition, 2108 STATUS:OPERation:PCALibration:ENABLE, 2108 STATUS:OPERation:PCALibration:EVENT, 2109 STATUS:OPERation:PCALibration:NTRansition, 2110 STATUS:OPERation:PCALibration:PTRansition, 2110 STATUS:OPERation:PTRansition, 2111 STATUS:PRESet, 2105 STATUS:QUESTionable:ACPLimit:CONDition, 2112 STATUS:QUESTionable:ACPLimit:ENABLE, 2113 STATUS:QUESTionable:ACPLimit:EVENT, 2114 STATUS:QUESTionable:ACPLimit:NTRansition, 2114 STATUS:QUESTionable:ACPLimit:PTRansition, 2115 STATUS:QUESTionable:CALibration:CONDition, 2116 STATUS:QUESTionable:CALibration:ENABLE, 2116 STATUS:QUESTionable:CALibration:EVENT, 2117 STATUS:QUESTionable:CALibration:NTRansition, 2117 STATUS:QUESTionable:CALibration:PTRansition, 2118 STATUS:QUESTionable:CONDition, 2119
SOURCE:CURRENT:AUX:LIMit:HIGH, 311, 2063		
SOURCE:CURRENT:CONTRol<Source>:LIMit:HIGH, 312, 2064		
SOURCE:CURRENT:POWer<Source>:LIMit:HIGH, 313, 2066		
SOURCE:CURRENT:SEQUence:RESult, 314, 2067		
SOURCE:EXTERNAL:FREQUENCY:FACTor:DENominator, 315		
SOURCE:EXTERNAL:FREQUENCY:FACTor:NUMerator, 316		
SOURCE:EXTERNAL:FREQUENCY:OFFSet<FreqOffset>, 317		
SOURCE:EXTERNAL:POWer:LEVel, 318		
SOURCE:EXTERNAL:ROSCillator:SOURce, 319		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY, 2068		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:COUPling:STATe, 2069		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:FACTor:DENominator, 2070		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:FACTor:NUMerator, 2071		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:OFFSet, 2071		
SOURCE:EXTERNAL<ExternalGen>:FREQUENCY:SWEep:STATe, 2072		
SOURCE:EXTERNAL<ExternalGen>:POWer:LEVel, 2073		
SOURCE:EXTERNAL<ExternalGen>:STATe, 2077		
SOURCE:EXTERNAL<ExternalRosc>:ROSCillator:EXTERNAL:FREQUENCY, 2074		
SOURCE:EXTERNAL<ExternalRosc>:ROSCillator:EXTERNAL:FREQUENCY:MODE, 2075		
SOURCE:EXTERNAL<ExternalRosc>:ROSCillator:SOURce:STATe, 2076		
SOURCE:GENERator:CHANnel:COUPling, 320, 2078		
SOURCE:GENERator:DUTBypass, 321, 2079		
SOURCE:GENERator:FREQUENCY, 321, 2079		
SOURCE:GENERator:FREQUENCY:STEP, 2080		
SOURCE:GENERator:LEVel, 322, 2081		
SOURCE:GENERator:MODulation, 322, 2081		
SOURCE:GENERator:PULSe:PERiod, 323, 2082		
SOURCE:GENERator:PULSe:TRIGger:OUTPut, 324, 2083		
SOURCE:GENERator:PULSe:WIDTh, 325, 2084		
SOURCE:GENERator:STATe, 325, 2085		
SOURCE:NSource:STATe, 326		
SOURCE:POWer:LEVel:IMMediate:OFFSet, 2086		
SOURCE:POWer:SEQUence:RESult, 327, 2087		
SOURCE:TEMPerature:FRONTend, 2088		
SOURCE:VOLTage:AUX:LEVel:AMPLitude, 328, 2089		
SOURCE:VOLTage:AUX:LEVel:LIMit:HIGH, 329, 2090		
SOURCE:VOLTage:AUX:LEVel:LIMit:LOW, 330, 2091		

STATUS:QUESTionable:CORRection:CONDition, 2119	2144
STATUS:QUESTionable:CORRection:ENABle, 2120	STATUS:QUESTionable:INTegrity:UNCalibrated:CONDition, 2145
STATUS:QUESTionable:CORRection:EVENTt, 2121	STATUS:QUESTionable:INTegrity:UNCalibrated:ENABle, 2145
STATUS:QUESTionable:CORRection:NTRansition, 2121	STATUS:QUESTionable:INTegrity:UNCalibrated:EVENTt, 2146
STATUS:QUESTionable:CORRection:PTRansition, 2122	STATUS:QUESTionable:INTegrity:UNCalibrated:NTRansition, 2146
STATUS:QUESTionable:DIQ:CONDition, 2123	STATUS:QUESTionable:INTegrity:UNCalibrated:PTRansition, 2147
STATUS:QUESTionable:DIQ:ENABle, 2123	STATUS:QUESTionable:LIMit<Window>:CONDition, 2148
STATUS:QUESTionable:DIQ:EVENTt, 2124	STATUS:QUESTionable:LIMit<Window>:ENABle, 2149
STATUS:QUESTionable:DIQ:NTRansition, 2124	STATUS:QUESTionable:LIMit<Window>:EVENTt, 2150
STATUS:QUESTionable:DIQ:PTRansition, 2125	STATUS:QUESTionable:LIMit<Window>:NTRansition, 2150
STATUS:QUESTionable:ENABle, 2126	STATUS:QUESTionable:LIMit<Window>:PTRansition, 2151
STATUS:QUESTionable:EVENTt, 2126	STATUS:QUESTionable:LMARgin<Window>:CONDition, 2153
STATUS:QUESTionable:EXTended:CONDition, 2127	STATUS:QUESTionable:LMARgin<Window>:ENABle, 2153
STATUS:QUESTionable:EXTended:ENABle, 2127	STATUS:QUESTionable:LMARgin<Window>:EVENTt, 2154
STATUS:QUESTionable:EXTended:EVENTt, 2128	STATUS:QUESTionable:LMARgin<Window>:NTRansition, 2155
STATUS:QUESTionable:EXTended:INFO:CONDition, 2129	STATUS:QUESTionable:LMARgin<Window>:PTRansition, 2156
STATUS:QUESTionable:EXTended:INFO:ENABle, 2129	STATUS:QUESTionable:MODulation<Window>:CFrequency:CONDition, 1075
STATUS:QUESTionable:EXTended:INFO:EVENTt, 2130	STATUS:QUESTionable:MODulation<Window>:CFrequency:ENABle, 1076
STATUS:QUESTionable:EXTended:INFO:NTRansition, 2131	STATUS:QUESTionable:MODulation<Window>:CFrequency:EVENTt, 1077
STATUS:QUESTionable:EXTended:INFO:PTRansition, 2132	STATUS:QUESTionable:MODulation<Window>:CFrequency:NTRansition, 1077
STATUS:QUESTionable:EXTended:NTRansition, 2133	STATUS:QUESTionable:MODulation<Window>:CFrequency:PTRansition, 1078
STATUS:QUESTionable:EXTended:PTRansition, 2133	STATUS:QUESTionable:MODulation<Window>:CONDition, 1079
STATUS:QUESTionable:FREquency:CONDition, 2135	STATUS:QUESTionable:MODulation<Window>:ENABle, 1080
STATUS:QUESTionable:FREquency:ENABle, 2135	STATUS:QUESTionable:MODulation<Window>:EVENTt, 1081
STATUS:QUESTionable:FREquency:EVENTt, 2136	STATUS:QUESTionable:MODulation<Window>:EVM:CONDition, 1082
STATUS:QUESTionable:FREquency:NTRansition, 2136	STATUS:QUESTionable:MODulation<Window>:EVM:ENABle, 1082
STATUS:QUESTionable:FREquency:PTRansition, 2137	STATUS:QUESTionable:MODulation<Window>:EVM:EVENTt, 1083
STATUS:QUESTionable:INTegrity:CONDition, 2138	STATUS:QUESTionable:MODulation<Window>:EVM:NTRansition, 1084
STATUS:QUESTionable:INTegrity:ENABle, 2139	
STATUS:QUESTionable:INTegrity:EVENTt, 2139	
STATUS:QUESTionable:INTegrity:NTRansition, 2140	
STATUS:QUESTionable:INTegrity:PTRansition, 2141	
STATUS:QUESTionable:INTegrity:SIGNAL:CONDition, 2142	
STATUS:QUESTionable:INTegrity:SIGNAL:ENABle, 2142	
STATUS:QUESTionable:INTegrity:SIGNAL:EVENTt, 2143	
STATUS:QUESTionable:INTegrity:SIGNAL:NTRansition, 2143	
STATUS:QUESTionable:INTegrity:SIGNAL:PTRansition, 2144	

STATUS:QUESTionable:MODulation<Window>:EVM:PTRansition, 1085	2162	STATUS:QUESTionable:POWER:DCPNoise:ENABLE, 1086	2162
STATUS:QUESTionable:MODulation<Window>:FSK:CONDition, 1086	2163	STATUS:QUESTionable:POWER:DCPNoise:EVENT, 1086	2163
STATUS:QUESTionable:MODulation<Window>:FSK:ENABLE, 1086	2164	STATUS:QUESTionable:POWER:DCPNoise:NTRansition, 1087	2164
STATUS:QUESTionable:MODulation<Window>:FSK:EVENT, 1087	2165	STATUS:QUESTionable:POWER:DCPNoise:PTRansition, 1088	2165
STATUS:QUESTionable:MODulation<Window>:FSK:NTRansition, 1088	2166	STATUS:QUESTionable:POWER:ENABLE, 1089	2167
STATUS:QUESTionable:MODulation<Window>:FSK:PTRansition, 1089	2167	STATUS:QUESTionable:POWER:EVENT, 1090	2167
STATUS:QUESTionable:MODulation<Window>:FSK:NTRansition, 1090	2168	STATUS:QUESTionable:POWER:NTRansition, 1091	2168
STATUS:QUESTionable:MODulation<Window>:IQRHo:CONDition, 1091	2169	STATUS:QUESTionable:POWER:PTRansition, 1092	2170
STATUS:QUESTionable:MODulation<Window>:IQRHo:ENABLE, 1092	2170	STATUS:QUESTionable:SYNC:CONDition, 1093	2171
STATUS:QUESTionable:MODulation<Window>:IQRHo:EVENT, 1093	2171	STATUS:QUESTionable:SYNC:ENABLE, 1094	2172
STATUS:QUESTionable:MODulation<Window>:IQRHo:PTRansition, 1094	2172	STATUS:QUESTionable:SYNC:EVENT, 1095	2173
STATUS:QUESTionable:MODulation<Window>:IQRHo:NTRansition, 1095	2173	STATUS:QUESTionable:SYNC:NTRansition, 1096	2174
STATUS:QUESTionable:MODulation<Window>:IQRHo:PTRansition, 1096	2174	STATUS:QUESTionable:SYNC:PTRansition, 1097	2175
STATUS:QUESTionable:MODulation<Window>:IQRHo:NTRansition, 1097	2175	STATUS:QUESTionable:TEMPerature:CONDition, 1098	2176
STATUS:QUESTionable:MODulation<Window>:IQRHo:PTRansition, 1098	2176	STATUS:QUESTionable:TEMPerature:ENABLE, 1099	2177
STATUS:QUESTionable:MODulation<Window>:MAGNitude:CONDition, 1099	2177	STATUS:QUESTionable:TEMPerature:EVENT, 1100	2178
STATUS:QUESTionable:MODulation<Window>:MAGNitude:ENABLE, 1100	2178	STATUS:QUESTionable:TEMPerature:NTRansition, 1101	2179
STATUS:QUESTionable:MODulation<Window>:MAGNitude:PTRansition, 1101	2179	STATUS:QUESTionable:TEMPerature:PTRansition, 1102	2180
STATUS:QUESTionable:MODulation<Window>:MAGNitude:EVENT, 1102	2180	STATUS:QUESTionable:TIME:CONDition, 1103	2181
STATUS:QUESTionable:MODulation<Window>:MAGNitude:PTRansition, 1103	2181	STATUS:QUESTionable:TIME:ENABLE, 1104	2182
STATUS:QUESTionable:MODulation<Window>:MAGNitude:NTRansition, 1104	2182	STATUS:QUESTionable:TIME:EVENT, 1105	2183
STATUS:QUESTionable:MODulation<Window>:MAGNitude:PTRansition, 1105	2183	STATUS:QUESTionable:TIME:NTRansition, 1106	2184
STATUS:QUESTionable:MODulation<Window>:MAGNitude:PTRansition, 1106	2184	STATUS:QUESTionable:TIME:PTRansition, 1107	2185
STATUS:QUESTionable:MODulation<Window>:NTRansition, 1107	2185	STATUS:QUESTionable:TRANSDUCER:CONDition, 1108	2186
STATUS:QUESTionable:MODulation<Window>:PHASe:CONDition, 1108	2186	STATUS:QUESTionable:TRANSDUCER:ENABLE, 1109	2187
STATUS:QUESTionable:MODulation<Window>:PHASe:ENABLE, 1109	2187	STATUS:QUESTionable:TRANSDUCER:EVENT, 1110	2188
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1110	2188	STATUS:QUESTionable:TRANSDUCER:NTRansition, 1111	2189
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1111	2189	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1112	2190
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1112	2190	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1113	2191
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1113	2191	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1114	2192
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1114	2192	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1115	2193
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1115	2193	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1116	2194
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1116	2194	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1117	2195
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1117	2195	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1118	2196
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1118	2196	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1119	2197
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1119	2197	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1120	2198
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1120	2198	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1121	2199
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1121	2199	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1122	2200
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1122	2200	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1123	2201
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1123	2201	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1124	2202
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1124	2202	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1125	2203
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1125	2203	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1126	2204
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1126	2204	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1127	2205
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1127	2205	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1128	2206
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1128	2206	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1129	2207
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1129	2207	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1130	2208
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1130	2208	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1131	2209
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1131	2209	STATUS:QUESTionable:TRANSDUCER:PTRansition, 1132	2210
STATUS:QUESTionable:MODulation<Window>:PHASe:PTRansition, 1132	2210	STATUS:QUESTionable	

Index 2323

SYSTem:SREcorder:SYNC, 2246
SYSTem:TEST:REDO, 2246
SYSTem:TEST:UNDO, 2247

T

target_auto_flushing (*ScpiLogger attribute*), 2289
TRACe:IQ:APCon:A, 124
TRACe:IQ:APCon:B, 125
TRACe:IQ:APCon:RESult, 126
TRACe:IQ:APCon:STATe, 126
TRACe:IQ:AVERAge:COUNt, 127
TRACe:IQ:AVERAge:STATe, 127
TRACe:IQ:BWIDth, 1106
TRACe:IQ:DATA:FORMat, 128
TRACe:IQ:DIQFilter, 129
TRACe:IQ:EGATe:GAP, 130
TRACe:IQ:EGATe:LENGth, 131
TRACe:IQ:EGATe:NOF, 131
TRACe:IQ:EGATe:STATe, 132
TRACe:IQ:EGATe:TYPE, 132
TRACe:IQ:EVAL, 133
TRACe:IQ:FILE:REPetition:COUNt, 134, 1107, 2253
TRACe:IQ:RLENgth, 135
TRACe:IQ:SRATe, 137
TRACe:IQ:STATe, 136
TRACe:IQ:TPISample, 137
TRACe:IQ:WBAND:MBWidth, 138, 1108
TRACe:IQ:WBAND:STATe, 139, 1109
TRACe:IQ:WFILter, 139
TRACe<Window>:COPY, 355, 390, 2248
TRACe<Window>:IQ:SCAPture:BOUNDary, 2254
TRACe<Window>:IQ:SCAPture:TSTamp:SSTart, 2255
TRACe<Window>:IQ:SCAPture:TSTamp:TRIGger, 2255
TRIGger:IQ:MASTer:SOURce, 2257
TRIGger:IQ:SENDER:SOURce, 2258
TRIGger:MASTer:PORT, 2259
TRIGger:SENDER:PORT, 2260
TRIGger:SEQuence:BBPower:HOLDoff, 2261
TRIGger:SEQuence:DTIME, 568, 720, 2262
TRIGger:SEQuence:HOLDoff:TIME, 568, 721, 2262
TRIGger:SEQuence:IFPower:HOLDoff, 569, 722, 2263
TRIGger:SEQuence:IFPower:HYSTeresis, 570, 722, 2264
TRIGger:SEQuence:LEVel:AM:ABSolute, 2265
TRIGger:SEQuence:LEVel:AM:RELative, 2266
TRIGger:SEQuence:LEVel:EXTernal<ExternalPort>, 571, 724, 2267
TRIGger:SEQuence:LEVel:FM, 2268
TRIGger:SEQuence:LEVel:IFPower, 572, 724, 2268
TRIGger:SEQuence:LEVel:IQPower, 573, 725, 2269
TRIGger:SEQuence:LEVel:PM, 2270
TRIGger:SEQuence:LEVel:RFPower, 573, 726, 2270

TRIGger:SEQuence:LEVel:VIDeo, 2271
TRIGger:SEQuence:OSCilloscope:COUPling, 2272
TRIGger:SEQuence:RFPower:HOLDoff, 2273
TRIGger:SEQuence:SLOPe, 574, 726, 2273
TRIGger:SEQuence:SOURce, 575, 727, 2274
TRIGger:SEQuence:TIME:RINTerval, 2275
TRIGger<TriggerPort>:SEQuence:DTIME, 357, 1110
TRIGger<TriggerPort>:SEQuence:HOLDoff:TIME, 358, 1111
TRIGger<TriggerPort>:SEQuence:IFPower:HOLDoff, 1112
TRIGger<TriggerPort>:SEQuence:IFPower:HYSTeresis, 1113
TRIGger<TriggerPort>:SEQuence:LEVel:EXTernal<ExternalPort>, 360, 1114
TRIGger<TriggerPort>:SEQuence:LEVel:IFPower, 1115
TRIGger<TriggerPort>:SEQuence:LEVel:IQPower, 1116
TRIGger<TriggerPort>:SEQuence:LEVel:RFPower, 1117
TRIGger<TriggerPort>:SEQuence:LEVel:VIDeo, 1117
TRIGger<TriggerPort>:SEQuence:RFPower:HOLDoff, 1118
TRIGger<TriggerPort>:SEQuence:SLOPe, 361, 1119
TRIGger<TriggerPort>:SEQuence:SOURce, 362
TRIGger<TriggerPort>:SEQuence:TIME:RINTerval, 1120

U

udp_port (*ScpiLogger attribute*), 2288
UNIT:ANGLE, 2277
UNIT:PMETER<PowerMeter>:POWER, 577, 2278
UNIT:PMETER<PowerMeter>:POWER:RATio, 578, 2279
UNIT:THD, 2280