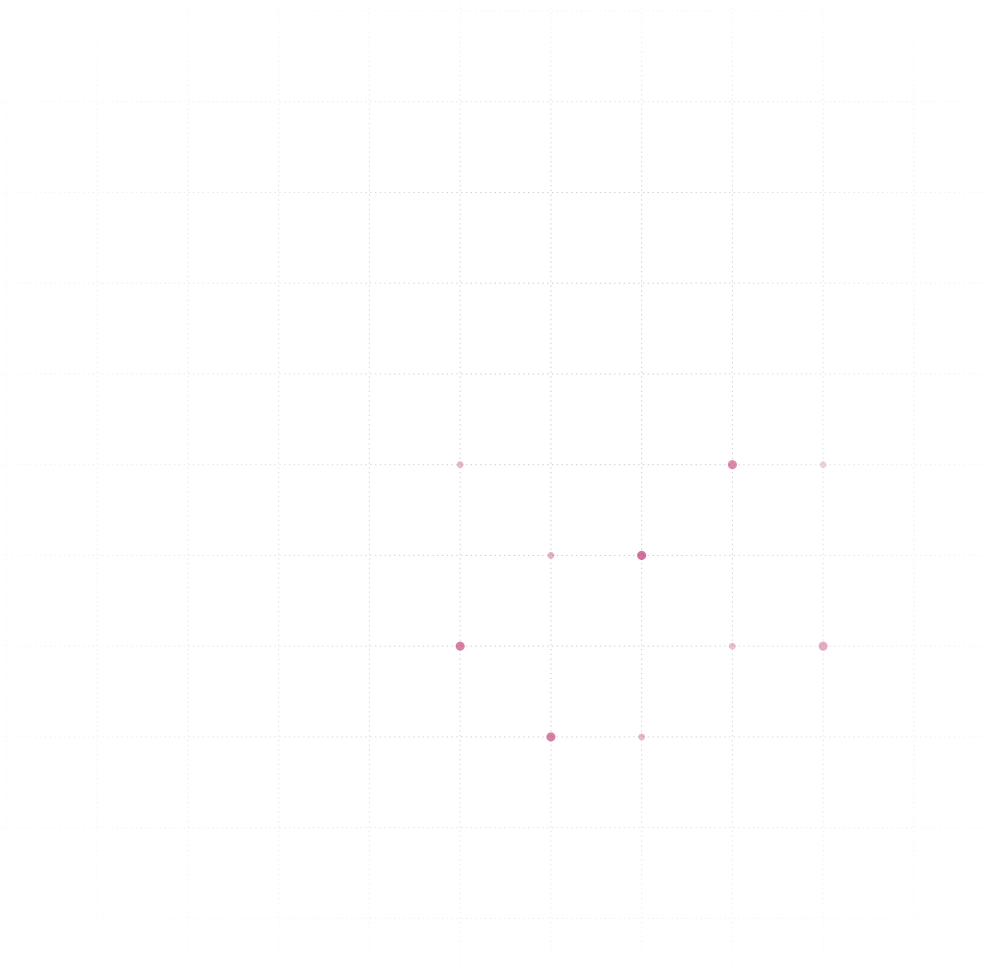
KS METROLOGY
SOLUTIONS

SCPI COMMAND REFERENCE

LPS Series

SCPI command set for remote control and automated test over USB and LAN



Contents

1 Remote control basics 1
- 1.1 Interfaces 1
- 1.2 SCPI message structure 2
- 1.3 Parameter types 3
- 1.4 Channel addressing 3
- 1.5 Synchronization and reset 4
- 1.6 Status system 4
  - 1.6.1 Status Byte 5
  - 1.6.2 Standard Event register 5
  - 1.6.3 Operation status 5
  - 1.6.4 Questionable status 6
  - 1.6.5 Measurement status 6
  - 1.6.6 Defaults and clearing 7
  - 1.6.7 Service request example 7
- 1.7 Error queue 7
- 1.8 Notation in this manual 8

2 IEEE 488.2 common commands 9
- 2.1 *CLS 10
- 2.2 *ESE 10
- 2.3 *ESR? 11
- 2.4 *IDN? 12
- 2.5 *OPC 13
- 2.6 *OPT? 14
- 2.7 *RCL 15
- 2.8 *RST 16
- 2.9 *SAV 16
- 2.10 *SRE 17
- 2.11 *STB? 18
- 2.12 *TRG 19
- 2.13 *TST? 20
- 2.14 *WAI 21

3 ABORt and INITiate commands 22
- 3.1 ABORt 22
- 3.2 INITiate 23

4 DIGital commands 26
- 4.1 Lines and functions 27

4.2 Conventions 28
4.3 DIGital:COUNter:CLEar 29
4.4 DIGital:COUNter:DATA? 30
4.5 DIGital:COUNter:SLOPe 30
4.6 DIGital:FAULt:ACTive 31
4.7 DIGital:FAULt:SOURce 32
4.8 DIGital:I2C:FREQuency 34
4.9 DIGital:I2C:READ? 35
4.10 DIGital:I2C:WRITe 37
4.11 DIGital:LINE<n>:MODE 39
4.12 DIGital:LINE<n>:READ? 40
4.13 DIGital:LINE<n>:STATe 41
4.14 DIGital:MODE 42
4.15 DIGital:PWM<i>:ACTive 44
4.16 DIGital:PWM<i>:DCYCle 45
4.17 DIGital:PWM<i>:FREQuency 46
4.18 DIGital:READ? 48
4.19 DIGital:SPI:CPHase 48
4.20 DIGital:SPI:CPOLarity 49
4.21 DIGital:SPI:CS<c>:ACTive 50
4.22 DIGital:SPI:FREQuency 51
4.23 DIGital:SPI:ORDer 53
4.24 DIGital:SPI:TRANsfer? 54
4.25 DIGital:SPI:WRITe 55
4.26 DIGital:STATe 57
4.27 DIGital:TINPut:SLOPe 58
4.28 DIGital:TOUTput:POLarity 59
4.29 DIGital:TOUTput:PWIDth 60
4.30 DIGital:UART:AVAilable? 62
4.31 DIGital:UART:BAUD 62
4.32 DIGital:UART:BITS 64
4.33 DIGital:UART:CLEar 65
4.34 DIGital:UART:ERRor:CLEar 66
4.35 DIGital:UART:ERRor? 66
4.36 DIGital:UART:PARity 67
4.37 DIGital:UART:READ? 68
4.38 DIGital:UART:SBITs 69
4.39 DIGital:UART:WRITe 70

5 DISPlay commands 72
5.1 DISPlay:BRIGhtness 72
5.2 DISPlay:STATe 74
5.3 DISPlay[:WINDow]:TEXT:CLEar 75
5.4 DISPlay[:WINDow]:TEXT[:DATA] 75

6 FETCh commands 77
6.1 FETCh:ARRay? 77
6.2 FETCh:ARRay:NEXT? 79
6.3 FETCh:ARRay:POINts? 80
6.4 FETCh[:SCALar]:CURRent[:DC]? 81
6.5 FETCh[:SCALar]:POWer? 82
6.6 FETCh[:SCALar]:VOLTage[:DC]? 83

7 FORMat commands 85
7.1 FORMat:BORDer 85
7.2 FORMat:SREGister 86
7.3 FORMat[:DATA] 88

8 HCOPy commands 90
8.1 HCOPy:DATA? 90

9 MEASure commands 92
9.1 MEASure:HISTogram:CURRent:BIN? 93
9.2 MEASure:HISTogram:CURRent:DATA? 94
9.3 MEASure:HISTogram:CURRent:EDGE? 96
9.4 MEASure:HISTogram:CURRent:POINts? 97
9.5 MEASure:HISTogram:POWer:BIN? 98
9.6 MEASure:HISTogram:POWer:DATA? 99
9.7 MEASure:HISTogram:POWer:EDGE? 100
9.8 MEASure:HISTogram:POWer:POINts? 101
9.9 MEASure:HISTogram:VOLTage:BIN? 102
9.10 MEASure:HISTogram:VOLTage:DATA? 103
9.11 MEASure:HISTogram:VOLTage:EDGE? 105
9.12 MEASure:HISTogram:VOLTage:POINts? 106
9.13 MEASure[:SCALar]:CURRent[:DC]? 107
9.14 MEASure[:SCALar]:CURRent[:DC]:AVERage? 108
9.15 MEASure[:SCALar]:CURRent[:DC]:MAXimum? 109
9.16 MEASure[:SCALar]:CURRent[:DC]:MINimum? 110
9.17 MEASure[:SCALar]:CURRent[:DC]:PTPeak? 111
9.18 MEASure[:SCALar]:ENERgy? 112
9.19 MEASure[:SCALar]:ENERgy:RESet 113
9.20 MEASure[:SCALar]:ENERgy:UNIT 114
9.21 MEASure[:SCALar]:POWer? 115
9.22 MEASure[:SCALar]:POWer:AVERage? 117
9.23 MEASure[:SCALar]:POWer:MAXimum? 118
9.24 MEASure[:SCALar]:POWer:MINimum? 119
9.25 MEASure[:SCALar]:POWer:PTPeak? 120
9.26 MEASure[:SCALar]:STATistic:COUNt? 121
9.27 MEASure[:SCALar]:STATistic:DURation? 122
9.28 MEASure[:SCALar]:STATistic:RESet 123

9.29 MEASure[:SCALar]:VOLTage[:DC]? 124
9.30 MEASure[:SCALar]:VOLTage[:DC]:AVERage? 125
9.31 MEASure[:SCALar]:VOLTage[:DC]:MAXimum? 126
9.32 MEASure[:SCALar]:VOLTage[:DC]:MINimum? 127
9.33 MEASure[:SCALar]:VOLTage[:DC]:PTPeak? 128

10 MMEMory commands 130
10.1 MMEMory:CAPacity? 130
10.2 MMEMory:DATA:APPend 132
10.3 MMEMory:DATA[:DEFinite] 133
10.4 MMEMory:DELete 135
10.5 MMEMory:FREE? 136
10.6 MMEMory:INITialize 138
10.7 MMEMory:LIST? 139

11 OUTPut commands 141
11.1 OUTPut:INHibit:ACTive 141
11.2 OUTPut:INHibit:MODE 142
11.3 OUTPut:MODE 144
11.4 OUTPut:PROTection:CLEar 145
11.5 OUTPut[:STATe] 146

12 SENSe commands 149
12.1 SENSe:CURRent:RANGe 150
12.2 SENSe:CURRent:RANGe:AUTO 152
12.3 SENSe:FUNCtion:CURRent 153
12.4 SENSe:FUNCtion:VOLTage 155
12.5 SENSe:SWEep:OFFSet:POINts 156
12.6 SENSe:SWEep:POINts 158
12.7 SENSe:SWEep:TINTerval 159
12.8 SENSe:VOLTage:RANGe 161
12.9 SENSe:VOLTage:RANGe:AUTO 163

13 SOURce commands 165
13.1 [SOURce:]ARB:EXPonential:END:TIME 168
13.2 [SOURce:]ARB:EXPonential:END[:LEVel] 169
13.3 [SOURce:]ARB:EXPonential:STARt:TIME 171
13.4 [SOURce:]ARB:EXPonential:STARt[:LEVel] 173
13.5 [SOURce:]ARB:EXPonential:TCONstant 174
13.6 [SOURce:]ARB:EXPonential:TIME 176
13.7 [SOURce:]ARB:FUNCtion:SHAPe 178
13.8 [SOURce:]ARB:LOGStaircase:END:TIME 179
13.9 [SOURce:]ARB:LOGStaircase:END[:LEVel] 181
13.10 [SOURce:]ARB:LOGStaircase:NSTeps 183
13.11 [SOURce:]ARB:LOGStaircase:STARt:TIME 184

13.12 [SOURce:]ARB:LOGStaircase:STARt[:LEVel] 186
13.13 [SOURce:]ARB:LOGStaircase:TIME 188
13.14 [SOURce:]ARB:MODE 189
13.15 [SOURce:]ARB:PULSe:HIGH 190
13.16 [SOURce:]ARB:PULSe:HIGH:TIME 192
13.17 [SOURce:]ARB:PULSe:LOW 194
13.18 [SOURce:]ARB:PULSe:LOW:TIME 195
13.19 [SOURce:]ARB:RAMP:END:TIME 197
13.20 [SOURce:]ARB:RAMP:END[:LEVel] 199
13.21 [SOURce:]ARB:RAMP:RTIMe 200
13.22 [SOURce:]ARB:RAMP:STARt:TIME 202
13.23 [SOURce:]ARB:RAMP:STARt[:LEVel] 204
13.24 [SOURce:]ARB:SEQuence:COUNt 205
13.25 [SOURce:]ARB:SEQuence:DATA 207
13.26 [SOURce:]ARB:SEQuence:DATA:POINts? 209
13.27 [SOURce:]ARB:SEQuence:FILE 210
13.28 [SOURce:]ARB:SEQuence:PAUSe 212
13.29 [SOURce:]ARB:SEQuence:SOURce 213
13.30 [SOURce:]ARB:SINusoid:AMPLitude 215
13.31 [SOURce:]ARB:SINusoid:FREQuency 217
13.32 [SOURce:]ARB:SINusoid:OFFSet 218
13.33 [SOURce:]ARB:SINusoid:PHASe 220
13.34 [SOURce:]ARB:STAircase:END:TIME 222
13.35 [SOURce:]ARB:STAircase:END[:LEVel] 223
13.36 [SOURce:]ARB:STAircase:NSTeps 225
13.37 [SOURce:]ARB:STAircase:STARt:TIME 227
13.38 [SOURce:]ARB:STAircase:STARt[:LEVel] 228
13.39 [SOURce:]ARB:STAircase:TIME 230
13.40 [SOURce:]ARB:STReam:DATA 232
13.41 [SOURce:]ARB:STReam:DATA:CURRent 233
13.42 [SOURce:]ARB:STReam:DATA:VOLTage 234
13.43 [SOURce:]ARB:STReam:POINts 236
13.44 [SOURce:]ARB:TERMinate:LAST 237
13.45 [SOURce:]ARB:TINTerval 238
13.46 [SOURce:]ARB:TRAPezoid:END:TIME 240
13.47 [SOURce:]ARB:TRAPezoid:FTIMe 241
13.48 [SOURce:]ARB:TRAPezoid:RTIMe 243
13.49 [SOURce:]ARB:TRAPezoid:STARt:TIME 245
13.50 [SOURce:]ARB:TRAPezoid:STARt[:LEVel] 246
13.51 [SOURce:]ARB:TRAPezoid:TOP:TIME 248
13.52 [SOURce:]ARB:TRAPezoid:TOP[:LEVel] 249
13.53 [SOURce:]ARB:TRIangle:FTIMe 251
13.54 [SOURce:]ARB:TRIangle:RTIMe 253
13.55 [SOURce:]ARB:TRIangle:STARt[:LEVel] 254

13.56 [SOURce:]ARB:TRIangle:TOP[:LEVel] 256
13.57 [SOURce:]ARB[:STATe] 258
13.58 [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] 259
13.59 [SOURce:]CURRent:PROTection:CLEar 261
13.60 [SOURce:]CURRent:PROTection:DELay 262
13.61 [SOURce:]CURRent:PROTection:LEVel 264
13.62 [SOURce:]CURRent:PROTection:TRIPped? 265
13.63 [SOURce:]CURRent:PROTection[:STATe] 266
13.64 [SOURce:]POWer:PROTection:CLEar 268
13.65 [SOURce:]POWer:PROTection:DELay 269
13.66 [SOURce:]POWer:PROTection:LEVel 270
13.67 [SOURce:]POWer:PROTection:TRIPped? 272
13.68 [SOURce:]POWer:PROTection[:STATe] 273
13.69 [SOURce:]VOLTage:PROTection:CLEar 274
13.70 [SOURce:]VOLTage:PROTection:LEVel 275
13.71 [SOURce:]VOLTage:PROTection:TRIPped? 277
13.72 [SOURce:]VOLTage:PROTection[:STATe] 278
13.73 [SOURce:]VOLTage:SENSe[:SOURce] 280
13.74 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude] 281

14 STATus commands 284
14.1 STATus:OPERation:CONDition? 285
14.2 STATus:OPERation:ENABle 285
14.3 STATus:OPERation:NTRansition 287
14.4 STATus:OPERation:PTRansition 288
14.5 STATus:OPERation[:EVENt]? 289
14.6 STATus:QUEStionable:CONDition? 290
14.7 STATus:QUEStionable:ENABle 291
14.8 STATus:QUEStionable:NTRansition 292
14.9 STATus:QUEStionable:PTRansition 293
14.10 STATus:QUEStionable[:EVENt]? 294
14.11 STATus:MEASurement:CONDition? 295
14.12 STATus:MEASurement:ENABle 295
14.13 STATus:MEASurement:NTRansition 297
14.14 STATus:MEASurement:PTRansition 298
14.15 STATus:MEASurement[:EVENt]? 299
14.16 STATus:PRESet 300
14.17 STATus:QUEue:CLEar 300
14.18 STATus:QUEue:DISable 301
14.19 STATus:QUEue:ENABle 302
14.20 STATus:QUEue[:NEXT]? 303

15 SYSTem commands 305
15.1 SYSTem:BEEPer:STATe 307

15.2 SYSTem:BEEPer[:IMMediate] 308
15.3 SYSTem:COMMunicate:LAN:APPLy 309
15.4 SYSTem:COMMunicate:LAN:DESCription 309
15.5 SYSTem:COMMunicate:LAN:DESCription:STATus? 311
15.6 SYSTem:COMMunicate:LAN:DISCard 311
15.7 SYSTem:COMMunicate:LAN:HOSTname 312
15.8 SYSTem:COMMunicate:LAN:HOSTname:STATus? 313
15.9 SYSTem:COMMunicate:LAN:IPV4:ADDRess 314
15.10 SYSTem:COMMunicate:LAN:IPV4:GATeway 315
15.11 SYSTem:COMMunicate:LAN:IPV4:MODE 316
15.12 SYSTem:COMMunicate:LAN:IPV4:STATus? 317
15.13 SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess? 318
15.14 SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway? 319
15.15 SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask? 319
15.16 SYSTem:COMMunicate:LAN:IPV4:SUBNetmask 320
15.17 SYSTem:COMMunicate:LAN:IPV6:ADDRess 321
15.18 SYSTem:COMMunicate:LAN:IPV6:GATeway 322
15.19 SYSTem:COMMunicate:LAN:IPV6:MODE 323
15.20 SYSTem:COMMunicate:LAN:IPV6:PREFix 324
15.21 SYSTem:COMMunicate:LAN:IPV6:STATus? 325
15.22 SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal? 326
15.23 SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal? 327
15.24 SYSTem:COMMunicate:LAN:MACaddress? 328
15.25 SYSTem:COMMunicate:LAN:RESet 328
15.26 SYSTem:COMMunicate:LXI:EVENt:DOMain 329
15.27 SYSTem:COMMunicate:RLSTate 330
15.28 SYSTem:COMMunicate:SELF? 331
15.29 SYSTem:COMMunicate:SERial:BAUDrate 332
15.30 SYSTem:COMMunicate:SERial:FLOWcontrol 333
15.31 SYSTem:COMMunicate:USB:CLASs 334
15.32 SYSTem:CSV:DECimal 335
15.33 SYSTem:CSV:DELimiter 336
15.34 SYSTem:CSV:LINeending 337
15.35 SYSTem:DATE 338
15.36 SYSTem:ERRor:ALL? 339
15.37 SYSTem:ERRor:CLEar 340
15.38 SYSTem:ERRor:CODE:ALL? 341
15.39 SYSTem:ERRor:CODE[:NEXT]? 341
15.40 SYSTem:ERRor:COUNt? 342
15.41 SYSTem:ERRor[:NEXT]? 343
15.42 SYSTem:LFRequency 344
15.43 SYSTem:LOCal 345
15.44 SYSTem:LOCK:OWNer? 346
15.45 SYSTem:REMote 346

15.46 SYSTem:RWLock 347
15.47 SYSTem:TIME 348
15.48 SYSTem:TIME:PTP:STATus? 349
15.49 SYSTem:TIME:SOURce? 350
15.50 SYSTem:TZONe 350
15.51 SYSTem:VERSion? 352

16 TRACe commands 353
16.1 TRACe:CLEar 353
16.2 TRACe:SAVE 354

17 TRIGger commands 356
17.1 TRIGger:CURRent:SLOPe 357
17.2 TRIGger:CURRent[:LEVel] 358
17.3 TRIGger:LXI:EVENt 359
17.4 TRIGger:TIME 361
17.5 TRIGger:VOLTage:SLOPe 362
17.6 TRIGger:VOLTage[:LEVel] 363
17.7 TRIGger[:SEQuence]:SOURce 365

A Reset and persistence 368
A.1 What *RST, *SAV, *RCL and power-up do 368
A.2 Reset values 369

About this document

This reference describes the native SCPI command set of the LPS Series. It is preliminary. The commands described here are stable. Battery simulation and the DVM input are not yet described, and values marked TBD are fixed with the released hardware. Chapter 1 explains how the instrument is addressed, how messages are built, how the status system reports events and how the command descriptions are to be read. The following chapters list the commands by subsystem.

Programmable voltage, current and power ranges are model constants: they are listed in the data sheet of the model, and the instrument returns them with the `MINimum` and `MAXimum` selectors. `*RST` and `DEFault` values are the same on every model and are listed in Appendix A.

1

CHAPTER 1

Remote control basics

This chapter covers what every program needs before the first command: the interfaces, the structure of SCPI messages, channel addressing, synchronization, the status system and the error queue. The last section explains the notation used in the command descriptions.

IN THIS CHAPTER

1.1	Interfaces	1
1.2	SCPI message structure	2
1.3	Parameter types	3
1.4	Channel addressing	3
1.5	Synchronization and reset	4
1.6	Status system	4
1.7	Error queue	7
1.8	Notation in this manual	8

1.1 Interfaces

The instrument accepts SCPI over three transports. All of them carry the same command set.

USB	USB Test and Measurement Class (TMC) or a virtual serial port (CDC). The class is selected with `SYSTem:COMMunicate:USB:CLASs` and is persistent.
LAN	HiSLIP, VXI-11 and a raw SCPI socket. The instrument announces itself through mDNS.
RS-232	Fixed 8N1 framing, baud rate and RTS/CTS flow control configurable with `SYSTem:COMMunicate:SERial:*`.

Each interface instance owns a session: its own error queue, Standard Event register, enable masks and *OPC bookkeeping. Several controllers can therefore monitor the instrument at the same time without consuming each other's events. Device conditions, settings and outputs are shared. `SYSTem:COMMunicate:SELF?` returns the interface of the current session.

1.2 SCPI message structure

Commands are organized in a tree. A command header is a path of mnemonics separated by colons, followed by parameters.

```
SOURce:VOLTage:PROTection:LEVel
  32,(@1)
```

Long and short form	Every mnemonic has a long form and a short form written in capitals: `VOLTage` is sent as `VOLT` or `VOLTAGE`. Any other truncation, such as `VOLTA`, is a command error. Case is ignored.
Default nodes	Nodes shown in square brackets may be omitted. `[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]` is sent as `VOLT`.
Parameters	Separated from the header by at least one space and from each other by commas.
Program message	Several commands in one message are separated by semicolons. After a semicolon the next header is relative to the previous path; a leading colon returns to the root. Common commands *XXX are independent of the path.
Terminator	A message ends with a line feed. USB TMC, VXI-11 and HiSLIP additionally mark the end of a message on the transport level.
Queries	A question mark after the header requests a response. The responses of several queries in one message are returned in one line, separated by semicolons, terminated by a line feed.

```
VOLT 5;CURR 2;OUTP ON
-> (no response)
:VOLT?;CURR?
-> 5.000000000E+00;2.000000000E+00
```

1.3 Parameter types

NRf	Decimal number with optional exponent, for example `5` or `2.5E-3`. A physical quantity accepts its SI unit with a decimal prefix as a suffix, for example `10 mV`, `500 mA` or `2.5 kHz`; without suffix the base unit of the command applies (V, A, W, s, Hz, degree). Dimensionless parameters such as counts, indices, masks and slot numbers take no suffix.
NR1	Integer. Also the response form of Boolean values: `0` or `1`.
NR3	Number in exponent form as returned by every non-integer numeric query: ten significant digits, no leading plus sign, for example `5.000000000E+00` or `-2.500000000E-03`. Overrange readings are returned as `9.900000000E+37` or `-9.900000000E+37`; statistics without a valid sample return `9.910000000E+37`.
CPD	Character program data: a mnemonic such as `MINimum` or `INTernal`, with long and short form like a command node.
CRD	Character response data: the short form returned by queries, for example `INT`.
Boolean	`ON`, `OFF`, `1` or `0`.
String	Text in double quotes.
Block	IEEE 488.2 definite-length block: #, one digit giving the length of the length field, the length in bytes, then the data.
AARD	Arbitrary ASCII response data: text without quotes, returned by `*IDN?` and `*OPT?`.

Numeric settings accept the selectors `MINimum`, `MAXimum` and `DEFault` instead of a number. The corresponding query with the same selector returns the limit or the default without changing the setting.

1.4 Channel addressing

Every channel-dependent command takes an optional channel list. The rules are the same on every model of the family; models differ only in the number of channels.

Omitted list	`VOLT 5` is `VOLT 5,(@1)` and `VOLT?` is `VOLT? (@1)`. There is no persistent channel selection.
Order	A list is expanded in the order written: `(@3,1,2)` addresses CH3, CH1, CH2; `(@3:1)` addresses CH3, CH2, CH1. A channel listed twice generates `-171,"Invalid expression"`.
Setting	One value is applied to every listed channel: `VOLT 5,(@1:3)`.
Simple query	One item per listed channel, comma-separated, in list order: `VOLT? (@3,1)` returns `<V3>,<V1>`.
Compound query	A query whose result is already a list accepts exactly one channel, shown as `(@<channel>)` in its syntax.

Event suffix	Trigger events that observe a channel carry a numeric suffix: `VOLTage<n>`, `CURRent<n>`, `CV<n>`, `CC<n>`, `OVP<n>`, `OCP<n>` and `OPP<n>`. Omitted = 1.
Unknown channel	A channel that does not exist on the model generates `-222,"Data out of range"` and changes nothing.

1.5 Synchronization and reset

Commands are executed in order within one session. Operations that take time, such as a triggered acquisition, run in the background; the following commands synchronize with them.

`*OPC?`	Waits until all pending operations have completed, then returns 1. Following commands in the same stream wait as well.
`*OPC`	Does not block. Sets the OPC bit in the Standard Event register when the pending operations complete, for use with service requests.
`*WAI`	Suspends the following commands until pending operations have completed. No response.
`*RST`	Resets the volatile configuration, switches all outputs off and aborts running operations. Does not clear the error queue or the status programming.
`*CLS`	Clears the event registers, the error queue and pending `*OPC` requests of the calling session. Does not change settings, condition registers or unread responses.
`*SAV`, `*RCL`	Save and recall the instrument configuration in slots 0 to 9. The output state is never recalled; outputs are off after `*RCL`.

1.6 Status system

The instrument uses the IEEE 488.2 Standard Event register together with three 16-bit SCPI status groups: `STATus:OPERation`, `STATus:QUEStionable` and `STATus:MEASurement`. Every group follows the same model.



Condition	The current state of the instrument. Reading it changes nothing.
Transition filter	Selects whether a rising (`PTRansition`) or falling (`NTRansition`) condition latches an event. Default: rising transitions only.
Event	Latched per session. Reading `EVENt?` returns and clears the register; the condition is not affected.

Enable	Selects which events propagate into the group's summary bit of the Status Byte. Default: none.
Service request	*SRE selects which Status Byte bits raise a service request.

1.6.1 Status Byte

BIT	MASK	NAME	MEANING
7	0x80	**OPER**	An enabled Operation event is latched.
6	0x40	**MSS/RQS**	Master summary (*STB?) or request for service (serial poll).
5	0x20	**ESB**	An enabled Standard Event bit is set.
4	0x10	**MAV**	A response is waiting to be read.
3	0x08	**QUES**	An enabled Questionable event is latched.
2	0x04	**EAV**	The error queue of this session is not empty.
1	0x02		Reserved, always 0.
0	0x01	**MEAS**	An enabled Measurement event is latched.

1.6.2 Standard Event register

BIT	MASK	NAME	MEANING
7	0x80	**PON**	Power on. Set in every session at power-up and kept until read with *ESR? or cleared with *CLS.
6	0x40	**URQ**	User request. No source is defined, normally 0.
5	0x20	**CME**	Command error (-1xx).
4	0x10	**EXE**	Execution error (-2xx).
3	0x08	**DDE**	Device-dependent error, including asynchronous hardware and data-path failures.
2	0x04	**QYE**	Query error (-4xx).
1	0x02	**RQC**	Request control. Not used, always 0.
0	0x01	**OPC**	Operation complete, set by a completed *OPC request.

1.6.3 Operation status

Live regulation and power-flow state. These are not faults. With the output off all bits are 0.

BIT	MASK	NAME	MEANING
0	0x0001	**CV**	The voltage loop determines the output.
1	0x0002	**CC**	The current limit has been reached; the output stays enabled in constant-current regulation.
2	0x0004	**SOURCING**	The output delivers power to the DUT.

3	0x0008	**SINKING**	The output absorbs power from the DUT.
4–15			Reserved, always 0.

1.6.4 Questionable status

Latched protection and fault conditions. OVP, OCP and OPP trips clear only through the corresponding `...:CLEar` command or a reset; OTP, INHIBIT and HW_FAULT clear only through `OUTPut:PROTection:CLEar` or *RST and only once their cause is gone, OTP once every channel is back in its normal temperature range and HW_FAULT once the hardware has recovered by itself; while the cause persists they stay set and a clear has no effect. In `LATChing` inhibit mode INHIBIT is set whenever the inhibit input is active, so `OUTPut:PROTection:CLEar` clears it only while the input is inactive, and *RST clears it because it also selects inhibit mode `OFF`; in `LIVE` inhibit mode INHIBIT simply follows the input. The remaining bits clear when a new operation of the same kind is started successfully. The register is device-wide: a bit is set while its condition is present on at least one channel; the affected channel is read with the `...:PROTection:TRIPped?` queries.

BIT	MASK	NAME	MEANING
0	0x0001	**OVP**	Overvoltage protection tripped.
1	0x0002	**OCP**	Overcurrent protection tripped.
2	0x0004	**OPP**	Overpower protection tripped.
3	0x0008	**OTP**	Overtemperature protection tripped; every output is off.
4	0x0010	**INHIBIT**	The external inhibit input disables the outputs; set whenever the input is active and latched in `LATChing` mode, following the input in `LIVE` mode.
5	0x0020	**HW_FAULT**	A hardware fault is latched: internal supply, channel control path or critical self-test; every output is off.
6	0x0040	**ACQ_OVERRUN**	Acquisition terminated because unread data would have been overwritten.
7	0x0080	**DAC_UNDERRUN**	Streamed waveform playback terminated because a point was not available in time.
8	0x0100	**STORAGE_ERROR**	A storage-backed operation failed. The cause is reported in the error queue.
9–15			Reserved, always 0.

1.6.5 Measurement status

Live state of the triggered operation and a generic overrange indication.

BIT	MASK	NAME	MEANING
0	0x0001	**WAITING_FOR_TRIGGER**	The operation is armed and waits for the selected trigger source.
1	0x0002	**ACTIVE**	The trigger was accepted and at least one participant is still running. Completion is the falling edge of this bit.
2	0x0004	**ADC_BUFFER_READY**	One `FETCh:ARRay:NEXT?` block can be returned immediately.

3	0x0008	**DAC_BUFFER_FREE**	One complete 4096-point stream block can be accepted immediately.
4	0x0010	**READING_OVERFLOW**	The last measurement point contained a reading outside the active range.
5–15			Reserved, always 0.

1.6.6 Defaults and clearing

Power-up	ESE, SRE and all ENABle masks are 0, PTRansition is 0xFFFF, NTRansition is 0, the error queue is empty and PON is set. Status programming is not retained across a power cycle.
STATus:PRESet	Restores the ENABle and transition masks of the three groups of the calling session to their power-up values. Clears no events.
***CLS**	Clears the event registers and the error queue of the calling session. Condition registers and masks are untouched.
***RST**	Resets the instrument configuration and the protection latches; OTP and HW_FAULT stay set while their cause persists. Does not clear event history or masks.

1.6.7 Service request example

Raise a service request on any protection trip, and a second one when a triggered operation completes:

```
*CLS
STAT:QUES:PTR 511;NTR 0;ENAB 511
STAT:MEAS:NTR 2;ENAB 2
*SRE 9
INIT ACQ
```

1.7 Error queue

Every session has its own error queue with at least 16 entries. SYSTem:ERRor[:NEXT]? returns and removes the oldest entry as <code>,"<message>", or 0,"No error". SYSTem:ERRor:COUNt? returns the number of pending entries. Errors caused by a command are queued in the session that sent it; asynchronous errors of an operation are queued in the session that started it.

-221	Settings conflict: the request conflicts with the current state or another setting.
-222	Data out of range: a numeric value is outside the supported range. The setting is unchanged.
-224	Illegal parameter value: syntactically valid, but not an allowed value for this command.

-230	Data corrupt or stale: the requested data are unavailable or no longer valid.
-240	Hardware error: the channel hardware is not reachable, for example the link to a channel controller is down. Every command that needs the channel hardware generates it.
-171	Invalid expression: a channel list names the same channel twice, directly or through overlapping ranges.
-350	Queue overflow: the queue was full when a further error occurred. The newest entry is replaced by this entry; older entries are kept.

Positive codes are device-specific errors: they report the behaviour of a device under test on the Digital-I/O buses or the content of a user file, not a fault of the instrument.

+300	I2C address not acknowledged: no device acknowledged the I2C address.
+301	I2C data not acknowledged: the device did not acknowledge a data byte of `DIGital:I2C:WRITe`.
+302	I2C bus error: arbitration lost, the bus stuck, or the clock stretched beyond the bus timeout.
+310	Sequence file syntax error: a row of an ARB sequence file is malformed, wrong field count, non-numeric field or unknown shape keyword. Nothing is armed.

1.8 Notation in this manual

Each command is documented as a topic with its setting form and its query form as separate blocks. A query is a command of its own and may take parameters.

[...]	Optional node or parameter: omit it, or give exactly one of the alternatives inside.
a|b	Alternatives without brackets: exactly one of them is required.
<...>	A value the program supplies, for example `<voltage>`.
(@<chanlist>)	Optional channel list, see section 1.4. `(@<channel>)` is exactly one channel.
Capitals	The short form of a mnemonic.
Errors	Error codes the command can generate beyond syntax errors. Reset values are listed in Appendix A.
Returns	Response format and meaning of the query form.
->	A typical response in an example.

2

CHAPTER 2

IEEE 488.2 common commands

This chapter describes the common commands for identification, reset, self-test, status clearing, completion synchronization, saved setups, the Standard Event register and the Status Byte with their enable masks, and the bus trigger.

IN THIS CHAPTER

2.1 *CLS 10
2.2 *ESE 10
2.3 *ESR? 11
2.4 *IDN? 12
2.5 *OPC 13
2.6 *OPT? 14
2.7 *RCL 15
2.8 *RST 16
2.9 *SAV 16
2.10 *SRE 17
2.11 *STB? 18
2.12 *TRG 19
2.13 *TST? 20
2.14 *WAI 21

2.1 *CLS

Clears the event registers and the error queue of the calling session.

All latched event registers and the error queue of the calling session are cleared, and pending *OPC and *OPC? requests of the calling session are cancelled: a cancelled *OPC does not set the OPC bit later, and a cancelled *OPC? does not return its 1 later. The running operations themselves continue, and the completion requests of other sessions are not affected. Condition registers, *ESE, *SRE, the enable masks and transition filters of the status groups, the instrument configuration and unread query responses are not changed. The Status Byte is re-evaluated from the remaining live conditions, and the RQS latch of the calling session is cleared.

COMMAND

```
*CLS
```

Example

```
*CLS
SYST:ERR?
-> 0,"No error"
```

Related commands

*ESR?, SYSTem:ERRor[:NEXT]?, STATus:PRESet, *RST

2.2 *ESE

Sets or queries which Standard Event bits are summarized in the Status Byte.

A bit set in the mask lets the corresponding bit of the Standard Event register contribute to the ESB summary bit 5 of the Status Byte, which *SRE can turn into a service request; the bit assignment is listed in chapter 1. Writing the mask neither clears the Standard Event register nor creates an event, and the summary is re-evaluated immediately, so a bit latched earlier starts contributing as soon as it is enabled. The mask belongs to the calling session and is not shared with other sessions; *CLS and STATus:PRESet leave it unchanged. The query is non-destructive and returns the value in the format selected by FORMat:SREGister.

COMMAND

```
*ESE
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 255	Standard Event bits that contribute to the ESB summary bit. Bits of the unused sources URQ and RQC are accepted and have no effect.

QUERY

```
*ESE?
```

Returns

<NR1> The current Standard Event enable mask, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 255. The mask is unchanged.

Example

```
*ESE 60
*ESE?
-> 60
```

Related commands

*ESR?, *SRE, *STB?, FORMat:SREGister

2.3 *ESR?

Returns and clears the Standard Event register.

The register latches the IEEE 488.2 events of the calling session: power on, command, execution, device-dependent and query errors, and the completion of an *OPC request; the bits are listed in chapter 1. The query returns the register and then clears exactly the bits it reported, so an event is delivered once and reading it never consumes the event history of another session. No other register is touched: *ESE, *SRE, the status groups, the condition registers and the error queue are unchanged, and the error that set CME, EXE, DDE or QYE remains readable with SYSTem:ERRor[:NEXT]?. PON is set in every session at power-up and stays latched until that session reads it here or clears it with *CLS; a reconnected transport is not a power-up and does not set it again.

QUERY

```
*ESR?
```

Returns

<NR1>	The latched Standard Event bits since the last read, in the format selected by FORMat:SREGister.

Example

```
*CLS
VOLT 1E9
*ESR?
-> 16
*ESR?
-> 0
```

Related commands

*ESE, *CLS, *STB?, SYSTem:ERRor[:NEXT]?

2.4 *IDN?

Returns the identification of the instrument.

The response is four comma-separated ASCII fields: manufacturer, model, serial number and firmware version. The manufacturer field is KS Metrology Solutions, the model field is the model designation, for example LPS3010, the firmware version has the form <major>.<minor>.<patch>. No field contains a comma, and the structure is never extended. The same four values are published byte-identically in the mDNS TXT record and in the LXI identification document, so *IDN? and LAN discovery always agree. No instrument state is changed.

QUERY

```
*IDN?
```

Returns

<AARD>	Four comma-separated ASCII fields without quotes: manufacturer, model, serial number, firmware version.

Example

```
*IDN?
-> KS Metrology Solutions,LPS3010,123456,1.0.0
```

Related commands

*OPT?, SYSTem:VERSion?

2.5 *OPC

Signals the completion of the preceding operations through the OPC bit, or waits for it and returns 1.

Both forms refer to the operations pending from commands accepted before *OPC, for example a triggered operation started with INITiate or stopped with ABORt; an operation is complete when every participant has finished, including the final flush and close of a DLog file. The command form does not block: following commands are processed immediately, the OPC bit of the Standard Event register is set when the pending operations complete, and commands accepted after *OPC do not delay that event. The query form waits until the pending operations have completed and then returns 1; it does not set the OPC bit, and following commands of the same session are held until it completes while other sessions are not blocked. *CLS cancels the pending requests of the calling session, *RST those of every session; a cancelled request neither sets the bit nor returns 1.

COMMAND

```
*OPC
```

QUERY

```
*OPC?
```

Returns

<NR1>	1 when the pending operations have completed.

Example

```
INIT ACQ;*OPC?
-> 1
*CLS
*ESE 1
INIT ACQ;*OPC
*ESR?
-> 1
```

Related commands

*WAI, *ESE, *ESR?, INITiate

2.6 *OPT?

Returns the installed instrument options.

The response is a comma-separated list of the installed user-visible options, or `0` when no option is installed. Option identifiers are stable product identifiers and do not depend on the firmware version. No instrument state is changed, and an instrument without options does not generate an error.

QUERY

```
*OPT?
```

Returns

<AARD>	Comma-separated list of the installed options, or `0` when none is installed.

Example

```
*OPT?
-> 0
```

Related commands

*IDN?

2.7 *RCL

Recalls the instrument configuration saved in a setup slot.

A running triggered operation is aborted and the affected subsystems return to idle. All outputs are switched off, then the source, measurement, protection, output-mode, trigger, data-format and Digital-I/O configuration of the slot is restored atomically. The Digital-I/O functions of the lines that lose them are stopped first, in the same order as by `DIGital:MODE`, then every other Digital-I/O value of the slot is applied, and the lines that receive a function are routed last, so a recalled `PWM` or `TOUTput` routing is active at once with the recalled values. The outputs stay off until enabled explicitly. The segment lists loaded with `[SOURce:]ARB:SEQuence:DATA` are not part of a slot and are empty after the recall. Latched protection trips, the status and event registers, the error queue and the other slots are not changed. When the recall cannot be completed, the outputs stay off and an execution error is queued.

COMMAND

```
*RCL
  <slot>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<slot>`	NR1	0 to 9	Setup slot to recall.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	`<slot>` is outside 0 to 9. Nothing is recalled.
-200	"Execution error"	The slot is empty, or the recall cannot be completed. The outputs stay off.

Example

```
*RCL 1
OUTP?
-> 0
```

Related commands

`*SAV`, `*RST`, `OUTPut[:STATe]`

2.8 *RST

Resets the volatile instrument configuration.

All outputs are switched off, a running triggered operation is aborted, and the source, measurement, protection, data-format and Digital-I/O settings return to the reset values listed in Appendix A. Volatile acquisition data and the statistics window are cleared, as are the OVP, OCP and OPP trip latches and the OTP, INHIBIT and HW_FAULT latches whose cause is gone; OTP and HW_FAULT stay set while their cause persists, and INHIBIT is cleared because the inhibit mode returns to OFF, after which the input no longer counts. Every Digital-I/O function is stopped first: an open I2C transaction is ended with STOP, the counters are cleared, the UART receive buffers and error flags discarded and a PWM stops at its inactive level; then every line is routed to GPIO and released. The inhibit input therefore has no effect until its routing, level and mode are programmed again; a controller that uses it as an interlock programs them after every *RST. Because the abort is device-wide, pending *OPC and *OPC? requests of every session are cancelled. The error queue, latched event registers, *ESE, *SRE and the enable masks of the status groups are not cleared; a Questionable event caused by a trip stays latched until read or cleared with *CLS. Saved setups, calibration, LAN, USB and RS-232 settings, clock and other persistent system settings are not changed.

COMMAND

```
*RST
```

Example

```
*RST
OUTP?
-> 0
```

Related commands

*CLS, *RCL, ABORt, STATus:PRESet

2.9 *SAV

Saves the current instrument configuration in a setup slot.

The configuration is stored atomically in nonvolatile memory and overwrites the previous contents of the slot. Saved are the settings that *RCL restores: source, measurement, protection, output mode, trigger, data-format and Digital-I/O configuration, the routing and the inhibit level and mode included; Appendix A lists them. Not saved are the output state, the segment lists loaded with [SOURce:]ARB:SEQuence:DATA, status and event registers, the error queue, unread responses, acquisition data, statistics, calibration, options, LAN, USB and RS-232 settings, date and time, and the instrument identity. The active configuration and the outputs are not changed.

COMMAND

```
*SAV
  <slot>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<slot>`	NR1	0 to 9	Setup slot to write.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<slot>` is outside 0 to 9. No slot is written.

Example

```
VOLT 5;CURR 2
*SAV 1
```

Related commands

*RCL, *RST

2.10 *SRE

Sets or queries which Status Byte bits generate a service request.

A bit set in the mask makes the corresponding Status Byte bit a service-request cause: while at least one enabled bit is 1, MSS in *STB? is 1 and a rise of MSS latches RQS for the transport's serial poll. Bit 6 itself is ignored and cannot enable MSS or RQS; it is always reported as 0 by the query, so *SRE 255 reads back as 191. The mask selects only what is reported; it neither latches nor clears events, and the group enables of STATus:OPERation, STATus:QUEStionable and STATus:MEASurement decide what reaches the summary bits in the first place. The mask belongs to the calling session; *CLS and STATus:PRESet leave it unchanged.

COMMAND

```
*SRE
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 255	Status Byte bits that generate a service request. Bit 6 and the reserved bit 1 are accepted and have no effect.

QUERY

```
*SRE?
```

Returns

<NR1> The current service-request enable mask with bit 6 reported as 0, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 255. The mask is unchanged.

Example

```
*SRE 40
*SRE?
-> 40
*SRE 255
*SRE?
-> 191
```

Related commands

*STB?, *ESE, STATus:QUEStionable:ENABle, STATus:MEASurement:ENABle

2.11 *STB?

Returns the Status Byte of the calling session.

The Status Byte summarizes the three status groups on bits 7, 3 and 0, the enabled Standard Event bits on ESB, the unread response data on MAV and the non-empty error queue on EAV; the bits are listed in chapter 1. In this query bit 6 is MSS and is 1 whenever any Status Byte bit enabled by *SRE is 1. The query is non-destructive: it clears no event register, no Standard Event bit, no error, no MAV state and not the RQS latch, and repeating it returns the same value as long as the underlying causes persist. The summary bits are computed from the live state of the calling session, so MAV, EAV, ESB and the group summaries of another session are not visible here. A transport-level serial poll returns the same byte but with RQS on bit 6, and servicing it clears only that latch.

QUERY

```
*STB?
```

Returns

<NR1>	The current Status Byte with MSS on bit 6, in the format selected by FORMat:SREGister.

Example

```
*CLS
*STB?
-> 0
VOLT 1E9
*STB?
-> 4
*ESE 16;*SRE 32
*STB?
-> 100
```

Related commands

*SRE, *ESR?, SYSTem:ERRor[:NEXT]?, STATus:QUEStionable[:EVENt]?

2.12 *TRG

Generates one bus trigger event.

An operation that is armed with TRIGger[:SEQuence]:SOURce BUS and waiting for its trigger accepts the event, which defines t0 for every participant; the operation is a one-shot consumer, so a further *TRG neither retriggers nor restarts it. If the operation accepts the event, the Digital-I/O line routed to TOUTput emits the pulse that marks t0, as for every accepted trigger; there is no separate bus route to Trigger Out. *TRG does not arm the operation and does not change the trigger configuration; with nothing armed on BUS it is a harmless no-op, produces no Trigger-Out pulse and generates no -211,"Trigger ignored". The out-of-band trigger operations of the transports, VXI-11 device_trigger, the USBTMC trigger and the HiSLIP trigger message, are bound to the same single event: they are delivered once to the armed operation, inject no text into the parser and create no response data.

COMMAND

```
*TRG
```

Example

```
TRIG:SOUR BUS
INIT ACQ
*TRG
*OPC?
-> 1
```

Related commands

`TRIGger[:SEQuence]:SOURce`, `INITiate`, `DIGital:LINE<n>:MODE`, `ABORt`

2.13 *TST?

Runs the internal self-test and returns its result.

The instrument is first put into the *RST state, so all outputs are off, then a real internal self-test is executed. No external load or measurement equipment is required. The query returns `0` when every test passes and `1` when at least one test fails; the failing tests are reported as device-dependent errors in the error queue of the calling session.

QUERY

```
*TST?
```

Returns

`<NR1>`	`0` when all tests pass, `1` when one or more tests fail.

Example

```
*TST?
-> 0
```

Related commands

`SYSTem:ERRor[:NEXT]?`, `*RST`

2.14 *WAI

Holds the following commands until the preceding operations have completed.

The commands after *WAI in the same session are not executed until all operations pending from the preceding commands have completed. The completion criteria are those of *OPC: an operation is complete when every participant has finished, including the final flush and close of a DLog file. No response is generated and no status bit is changed. Other sessions are not blocked.

COMMAND

```
*WAI
```

Example

```
INIT ACQ;*WAI;FETC:VOLT?
-> 5.000000000E+00
```

Related commands

*OPC, INITiate, FETCh[:SCALar]:VOLTage[:DC]?

3

CHAPTER 3

ABORt and INITiate commands

The instrument runs one triggered operation at a time. INITiate validates and arms its participants, the acquisition into the internal record, the acquisition into a file and the DAC playback, for one common trigger instant t0; ABORt returns the whole operation to Idle. The trigger event is selected with the TRIGger commands, the participants are configured with the SENSe and [SOURce:]ARB commands, and the state of the operation is reported in the Measurement status.

IN THIS CHAPTER

3.1 ABORt 22

3.2 INITiate 23

3.1 ABORt

Stops the armed or active triggered operation and returns it to Idle.

The stop is one atomic action for every participant, whether the operation is still waiting for its trigger or already active. The acquisition stops and keeps every point already recorded as a valid partial record; a DLog writes its buffered points, flushes and closes the current file segment, and the files written so far remain on the medium; DAC playback stops on every participating channel, restores the pre-playback voltage setpoint and current limit regardless of ARB:TERMinate:LAST, and clears the DAC sample buffer. ABORt is an overlapped command: it returns as soon as every participant has been told to stop, and the operation is complete only when the last participant has finished, for a DLog after the final flush and close of its file, and ACTIVE in the Measurement status stays 1 until then, and ACTIVE in the Measurement status stays 1 until then; during that tail a new INITiate generates -221, a second ABORt is a no-op, and *OPC, *OPC? and *WAI wait. Without a DLog participant there is no tail. ABORt changes no configuration, never re-enables an output and never clears a protection latch; while nothing is armed or active it is a harmless no-op. There is no per-participant abort: a running operation is stopped as a whole, and a shorter acquisition or playback is configured before INITiate. *RST and *RCL perform the same abort before applying their own semantics.

COMMAND

```
ABORt
```

Example

```
TRIG:SOUR BUS
INIT DLOG,"USB:/log.csv"
ABOR;*OPC?
-> 1
STAT:MEAS:COND?
-> 0
```

Related commands

`INITiate`, `STATus:MEASurement:CONDition?`, `*OPC?`, `[SOURce:]ARB[:STATe]`

3.2 INITiate

Arms the triggered operation with the listed participants for one common trigger instant.

The function list names the participants in any order: `ACQuire` and `DLOG` are mutually exclusive because there is one acquisition path, `TRANsient` combines with either, and a bare `INITiate` is `INITiate ACQuire`. The legal forms are `INITiate` and `INITiate ACQuire` (acquisition into the internal record), `INITiate TRANsient` (playback only, no record), `INITiate ACQuire,TRANsient` (record and playback), `INITiate DLOG,"<path>"` (acquisition into the file) and `INITiate DLOG,TRANsient,"<path>"` (file and playback). Validation is atomic: every participant is checked and prepared first, a failure of any one of them arms nothing, and only participating channels are validated. The sequence file of a participating channel with `SEQuence:SOURce FILE` is loaded completely during validation into the segment list of that channel, and a failed load leaves that list empty. While an operation is armed or active a further `INITiate` generates `-221` and does not restart, retrigger, extend or abort the running operation. A successful `INITiate` with `ACQuire` or `DLOG` discards the previous acquisition record, its consuming cursor and any unread remainder at once, without waiting for the trigger. The first event of the selected source defines t0 and later events of the same source are ignored; after t0 the participants run to their own ends independently of one another, and because `INITiate` is an overlapped command, `*OPC`, `*OPC?` and `*WAI` report completion only when every participant has finished, including the final flush and close of a DLog file.

COMMAND

```
INITiate
  [<function-list>][,"<path>"]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ACQuire`	CPD		Acquisition into the internal record with the current `SENSe:FUNCtion`, `SENSe:SWEep` and trigger settings. Omitted function list: `ACQuire`. Not combinable with `DLOG`.

DLOG	CPD	Acquisition into the file <path> instead of the internal record, with the same SENSe and trigger settings. Requires <path>; not combinable with ACQuire.
TRANsient	CPD	DAC playback on every channel with ARB[:STATe] ON, in the ARB:MODE and with the ARB configuration in effect at arm. Combinable with ACQuire or DLOG.
<path>	String	Destination file of the DLOG participant in the storage notation of the MMEMory commands; an omitted prefix means the external USB storage. Missing parent directories are created, an existing file is truncated, and the recording is continued in segments with a suffix before the extension, log_0001.csv, log_0002.csv, … for log.csv, before the 4 GiB file-size limit of the FAT file system is reached. Required with DLOG, not allowed without it.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation is already armed or active, or ABORt is still closing the DLog file of the previous one. The running operation is unchanged and nothing is armed.
-221	"Settings conflict"	The function list names ACQuire and DLOG together. Nothing is armed.
-221	"Settings conflict"	No SENSe:FUNCtion quantity is ON on any channel with ACQuire or DLOG; SENSe:SWEep:OFFSet:POINts is inconsistent with SENSe:SWEep:POINts, the pre-trigger history capacity or an IMMediate source; no channel is ARB[:STATe] ON with TRANsient; the trigger source is OVP<n>, OCP<n> or OPP<n> with TRANsient; the source is TIME and the target is not in the future or the system clock is invalid; a participating channel selects SEQuence:SOURce DATA without a loaded list or SEQuence:SOURce FILE with an empty path. Nothing is armed and no setting is changed.
-221	"Settings conflict"	With TRANsient, a participating channel has its output off, one execution of its waveform together with SEQuence:PAUSe lasts 0 s, or its native shape would leave the voltage capability of the model; in ARB:MODE STReam, STReam:POINts is 0 or fewer points than the smaller of 4096 and STReam:POINts have been preloaded. Nothing is armed and the DAC sample buffer is left as it is.
-221	"Settings conflict"	An overtemperature trip (OTP) or a hardware fault (HW_FAULT) is latched. Nothing is armed.
-221	"Settings conflict"	With TRANsient, the inhibit input is active or its latch is set (OUTPut:INHibit:MODE). Nothing is armed.
-222	"Data out of range"	A participating channel plays LOGStaircase with a start or end level of 0 V or with levels of opposite sign. Nothing is armed.
-222	"Data out of range"	A row of a sequence file carries a duration below one DAC update period or longer than 21 474.83647 s. Nothing is armed, and the segment list of that channel is left empty.
-223	"Too much data"	A sequence file of a participating channel has more than 4096 data rows. Nothing is armed, and the segment list of that channel is left empty.

-224	"Illegal parameter value"	The function list names a participant twice. Nothing is armed.
-250	"Mass storage error"	The DLog file or a sequence file cannot be prepared or read because of another file-system failure. Nothing is armed.
-251	"Missing mass storage"	<path> or a sequence file names a storage volume that is not available on the model. Nothing is armed.
-252	"Missing media"	No USB medium is present for <path> or a sequence file. Nothing is armed.
-253	"Corrupt media"	The medium of <path> or of a sequence file is unusable. Nothing is armed.
-254	"Media full"	The DLog file cannot be created because the medium has no free space. Nothing is armed.
-256	"File name not found"	The sequence file configured with ARB:SEQuence:FILE of a participating channel does not exist. Nothing is armed.
-257	"File name error"	<path> or the path of a sequence file is syntactically invalid or unsupported. Nothing is armed.
-258	"Media protected"	The storage or path of <path> is write-protected. Nothing is armed.
+310	"Sequence file syntax error"	A row of the sequence file of a participating channel is malformed: wrong field count, a non-numeric field or an unknown shape keyword. Nothing is armed, and the segment list of that channel is left empty.

Example

```
INIT;*OPC?
-> 1
OUTP ON
INIT ACQ,TRAN;*OPC?
-> 1
INIT DLOG,"USB:/log.csv";*OPC?
-> 1
```

Related commands

TRIGger[:SEQuence]:SOURce, SENSe:FUNCtion:VOLTage, [SOURce:]ARB[:STATe], STATus:MEASurement:CONDition?

4

CHAPTER 4

DIGital commands

This chapter describes the six digital lines of the rear-panel connector. Every line is routed to one function; the first section lists the functions and the lines that can carry them.

IN THIS CHAPTER

4.1 Lines and functions 27

4.2 Conventions 28

4.3 DIGital:COUNter:CLEar 29

4.4 DIGital:COUNter:DATA? 30

4.5 DIGital:COUNter:SLOPe 30

4.6 DIGital:FAULt:ACTive 31

4.7 DIGital:FAULt:SOURce 32

4.8 DIGital:I2C:FREQuency 34

4.9 DIGital:I2C:READ? 35

4.10 DIGital:I2C:WRITe 37

4.11 DIGital:LINE<n>:MODE 39

4.12 DIGital:LINE<n>:READ? 40

4.13 DIGital:LINE<n>:STATe 41

4.14 DIGital:MODE 42

4.15 DIGital:PWM<i>:ACTive 44

4.16 DIGital:PWM<i>:DCYCle 45

4.17 DIGital:PWM<i>:FREQuency 46

4.18 DIGital:READ? 48

4.19 DIGital:SPI:CPHase 48

4.20 DIGital:SPI:CPOLarity 49

4.21 DIGital:SPI:CS<c>:ACTive 50

4.22 DIGital:SPI:FREQuency 51

4.23 DIGital:SPI:ORDer 53

4.24 DIGital:SPI:TRANsfer? 54

4.25 DIGital:SPI:WRITe 55

4.26 DIGital:STATe 57

4.27 DIGital:TINPut:SLOPe 58

4.28 DIGital:TOUTput:POLarity 59

4.29 DIGital:TOUTput:PWIDth 60

4.30 DIGital:UART:AVAilable? 62

4.31 DIGital:UART:BAUD 62

4.32 DIGital:UART:BITS 64

4.33 DIGital:UART:CLEar 65

4.34 DIGital:UART:ERRor:CLEar 66

4.35 DIGital:UART:ERRor? 66

4.36 DIGital:UART:PARity 67

4.37 DIGital:UART:READ? 68

4.38 DIGital:UART:SBITs 69

4.39 DIGital:UART:WRITe 70

4.1 Lines and functions

Every line is an open-drain output with an internal pull-up. State `0` drives the line Low, state 1 releases it, and the pull-up pulls a released line High. The instrument never drives a line High, so High levels and rising edges depend on the pull-up and the load; the frequency and pulse-width limits of the data sheet apply without external load.

`STATe` is the stored output state of a line, `READ?` its measured level. The two differ whenever an external device drives a released line, and `READ?` works in every mode.

A line does nothing until `DIGital:LINE<n>:MODE` or `DIGital:MODE` routes it to a function. Routing is the enable: the function runs as soon as its line is routed and stops when the line returns to `GPIO`. Every mode value except `GPIO` is routed to one line at a time.

The functions:

- **GPIO**: a plain line with `STATe` and `READ?`; every line is `GPIO` after `*RST` and at power-on.
- **Trigger input** (`TINPut`): the selected edge triggers an operation with `TRIGger:SOURce EXTernal`.
- **Trigger output** (`TOUTput`): one pulse for every accepted trigger.
- **Fault output** (`FAULt`): asserted while a selected protection condition is present on any channel.

- **Inhibit input** (`INHibit`): disables the outputs; level and mode are set with `OUTPut:INHibit`.
- **PWM** (`PWM1` to `PWM3`): a periodic signal with programmable frequency and duty cycle.
- **Pulse counter** (`COUNter`): counts the selected edges with 64 bits.
- **UART** (`UART`): transmit and receive line; either may be routed alone.
- **SPI master** (`SPI`): needs `CLK` and `MOSI`, for `TRANsfer?` also `MISO`, and the chip select that a command names; `CS1` to `CS3` can be routed to any line.
- **I2C master** (`I2C`): needs SDA and SCL on any two lines; up to 100 kHz.

Only the transfer commands need their lines routed; the settings of an instance, `UART:CLEar` and `UART:ERRor?` are accepted at any time.

The table shows which line accepts which mode value. It is identical on every model of the series; any other combination generates `-224,"Illegal parameter value"`.

MODE VALUE	LINE 1	LINE 2	LINE 3	LINE 4	LINE 5	LINE 6
`GPIO`	×	×	×	×	×	×
`TINPut`	×	×	×	×	×	×
`TOUTput`		×			×	×
`FAULt`	×	×	×	×	×	×
`INHibit`	×	×	×	×	×	×
`PWM1`		×				
`PWM2`					×	
`PWM3`						×
`COUNter`	×					
`UART_TX`	×					
`UART_RX`		×				
`SPI_CLK`			×			
`SPI_MOSI`				×		
`SPI_MISO`					×	
`SPI_CS1`	×	×	×	×	×	×
`SPI_CS2`	×	×	×	×	×	×
`SPI_CS3`	×	×	×	×	×	×
`I2C_SDA`	×	×	×	×	×	×
`I2C_SCL`	×	×	×	×	×	×

4.2 Conventions

- `SLOPe` selects an edge, `POLarity` the level of a pulse and `ACTive` the level at which a function is asserted.
- <n> is the line number 1 to 6, <i> the PWM instance 1 to 3 and <c> the chip select 1 to 3.
- `PWM` without digit is the same as `PWM1`, as header and as mode value. Queries return the PWM instance with its digit.

- The pulse counter, the UART, the SPI master and the I2C master exist once and are written without an instance digit: `DIGital:UART:BAUD`, `UART_TX`. The suffix 1 is accepted as the same header or mode value, `DIGital:UART1:BAUD` is `DIGital:UART:BAUD` and `UART1_TX` is `UART_TX`; queries return the form without the digit. Any other suffix, such as `UART2`, names an instance that does not exist.
- Mode values with an underscore, such as `I2C_SDA`, have no short form and are always written and returned in full.
- Every setting of this chapter, the routing included, is volatile: *RST and power-on reset it. Appendix A lists the values.

4.3 DIGital:COUNter:CLEar

Sets the count of a pulse counter to 0.

The count is set to `0` atomically; no edge is lost across the clear. The slope and the routing are unchanged, and counting continues.

COMMAND

```
DIGital:COUNter:CLEar
```

Errors

CODE	MESSAGE	CONDITION
-113	`"Undefined header"`	The header names an instance that does not exist, such as `DIGital:COUNter2:CLEar`. No count is cleared.

Example

```
DIG:COUN:DATA?
-> 1520
DIG:COUN:CLE
DIG:COUN:DATA?
-> 0
```

Related commands

`DIGital:COUNter:DATA?`, `DIGital:COUNter:SLOPe`

4.4 DIGital:COUNter:DATA?

Returns the count of a pulse counter.

The count is the number of selected edges since the line was routed to COUNter or since the last clear. Reading does not change it. The counter wraps to 0 after $2^{64} - 1$; it is set to 0 by COUNter:CLEar and when its line loses the COUNter function.

QUERY

```
DIGital:COUNter:DATA?
```

Returns

<NR1>	The count, 0 to $2^{64} - 1$.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:COUNter2:DATA?. No response is returned.

Example

```
DIG:LINE1:MODE COUN
DIG:COUN:DATA?
-> 1520
```

Related commands

DIGital:COUNter:CLEar, DIGital:COUNter:SLOPe, DIGital:LINE<n>:MODE

4.5 DIGital:COUNter:SLOPe

Selects or queries the edge that a pulse counter counts.

A line routed to COUNter counts the selected edges from the moment it is routed. Changing the slope does not clear the count. Counting both edges is not offered. The maximum input frequency and the minimum pulse width are specified in the data sheet.

COMMAND

```
DIGital:COUNter:SLOPe
  POSitive|NEGative
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
POSitive	CPD		Count rising edges.
NEGative	CPD		Count falling edges.

QUERY

```
DIGital:COUNter:SLOPe?
```

Returns

<CRD> POS or NEG.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:COUNter2:SLOPe. The slope is unchanged.

Example

```
DIG:COUN:SLOP NEG
DIG:COUN:SLOP?
-> NEG
```

Related commands

DIGital:COUNter:DATA?, DIGital:COUNter:CLEar, DIGital:LINE<n>:MODE

4.6 DIGital:FAULt:ACTive

Selects or queries the level at which the fault output is asserted.

The line routed to FAULt is a level output, not a pulse: it is asserted for as long as any condition selected with FAULt:SOURce is present on any channel. With HIGH the line is released, High through the pull-up, while a fault is present and driven Low while none is, so an unpowered or disconnected instrument also reads as a fault; LOW is the opposite.

COMMAND

```
DIGital:FAULt:ACTive
  HIGH|LOW
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
HIGH	CPD		Released while a fault is present, driven Low while none is.
LOW	CPD		Driven Low while a fault is present, released while none is.

QUERY

```
DIGital:FAULt:ACTive?
```

Returns

<CRD> HIGH or LOW.

Example

```
DIG:LINE3:MODE FAUL
DIG:FAUL:ACT LOW
DIG:FAUL:ACT?
-> LOW
```

Related commands

DIGital:FAULt:SOURce, DIGital:LINE<n>:MODE, STATus:QUEStionable:CONDition?

4.7 DIGital:FAULt:SOURce

Selects or queries the fault conditions that assert the fault output.

The names are the Questionable status conditions of the protection functions. The fault output follows their condition bits over all channels: it is asserted while at least one selected condition is set on at least one channel and deasserted once the last selected condition is cleared on every channel. It has no latch and no acknowledgement of its own; the conditions are cleared with OUTPut:PROTection:CLEar and the other protection clear commands, and OTP, INHIBIT and HW_FAULT stay set while their cause persists, a clear has no effect then. A name given twice counts once. The pin is a report, not the protection: the protection logic switches the output off before the fault output can report it, and the reaction time is a product specification.

COMMAND

```
DIGital:FAULt:SOURce
  <source>[,<source>...]|ALL|NONE
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<source>`	CPD		One of the six condition names below; several names are separated by commas.
`OVP`	CPD		Overvoltage-protection trip, Questionable bit 0.
`OCP`	CPD		Overcurrent-protection trip, Questionable bit 1.
`OPP`	CPD		Overpower-protection trip, Questionable bit 2.
`OTP`	CPD		Overtemperature protection, Questionable bit 3.
`INHibit`	CPD		External inhibit, Questionable bit 4, `INHIBIT`.
`HWFault`	CPD		Hardware fault, Questionable bit 5, `HW_FAULT`.
`ALL`	CPD		All six conditions. Cannot be combined with a name.
`NONE`	CPD		No condition; the fault output stays deasserted. Cannot be combined with a name.

QUERY

```
DIGital:FAULt:SOURce?
```

Returns

`<CRD>[,<CRD>...]` The selected names in short form in the fixed order `OVP,OCP,OPP,OTP,INH,HWF`; `ALL` when all six are selected, `NONE` when none is.

Errors

CODE	MESSAGE	CONDITION
-224	`"Illegal parameter value"`	`ALL` or `NONE` is combined with a condition name. The selection is unchanged.

Example

```
DIG:FAUL:SOUR INH,OCP,OVP
DIG:FAUL:SOUR?
-> OVP,OCP,INH
DIG:FAUL:SOUR ALL
DIG:FAUL:SOUR?
-> ALL
```

Related commands

`DIGital:FAULt:ACTive`, `STATus:QUEStionable:CONDition?`, `OUTPut:PROTection:CLEar`

4.8 DIGital:I2C:FREQuency

Sets or queries the clock frequency of an I2C master.

The master is implemented in software; its range of 10 kHz to 100 kHz is the same on every model and does not depend on the pull-up. The clock is quantized, and the query returns the frequency actually programmed. The bus uses the external pull-ups of the application together with the internal ones, and the rising edges of the clock are produced by the pull-ups.

COMMAND

```
DIGital:I2C:FREQuency
  <hertz>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<hertz>`	NRf, Hz	10 kHz to 100 kHz	Clock frequency.
`MINimum`	CPD		Lowest programmable frequency, 10 kHz.
`MAXimum`	CPD		Highest programmable frequency, 100 kHz.
`DEFault`	CPD		Factory default, 100 kHz.

QUERY

```
DIGital:I2C:FREQuency?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable frequency, 10 kHz, instead of the setting.
`MAXimum`	CPD		Return the highest programmable frequency, 100 kHz, instead of the setting.
`DEFault`	CPD		Return the factory default, 100 kHz, instead of the setting.

Returns

`<NR3>, Hz` The frequency actually programmed after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<hertz>` is outside 10 kHz to 100 kHz. The frequency is unchanged.
-113	`"Undefined header"`	The header names an instance that does not exist, such as `DIGital:I2C2:FREQuency`. The frequency is unchanged.

Example

```
DIG:I2C:FREQ 50 kHz
DIG:I2C:FREQ?
-> 5.000000000E+04
DIG:I2C:FREQ? DEF
-> 1.000000000E+05
```

Related commands

`DIGital:I2C:WRITe`, `DIGital:I2C:READ?`, `DIGital:LINE<n>:MODE`

4.9 DIGital:I2C:READ?

Addresses an I2C device for reading and returns the received bytes.

The device is addressed with its unshifted 7-bit address and read with an exact byte count or, with AUTO, as an SMBus Block Read whose first byte is the byte count; `<extra>` reads further bytes such as an SMBus PEC byte, which the instrument neither computes nor checks. The transaction rules of `I2C:WRITe` apply: consecutive I2C commands on the same instance within one program message form one transaction, joined by Repeated START and ended by STOP at the end of the message. `DIG:I2C:WRIT 80,0,16;READ? 80,4` is therefore the random read of an EEPROM: START, address 80 write, bytes 0 and 16, Repeated START, address 80 read, 4 bytes, STOP; as two program messages, the two commands are two transactions with a STOP between them. An error ends the transaction with STOP, and the next I2C command of the program message starts a new one; Device Clear and the loss of the connection abort a running read at once with STOP. The response is always a definite-length

block, whatever FORMat[:DATA] and FORMat:BORDer are set to; a failed or aborted read returns no response data. The query completes when the bus activity it causes has ended.

QUERY

```
DIGital:I2C:READ?
  <address>,<count>|AUTO[,<extra>][,STOP]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<address>	NR1	0 to 127	Unshifted 7-bit address of the device.
<count>	NR1	1 to 32768	Exact number of bytes to read.
AUTO	CPD		SMBus Block Read: the first byte is read as the byte count N, then N further bytes are read. The count byte is part of the returned data.
<extra>	NR1	0 to 255	Additional bytes read after the counted ones, for example 1 for an SMBus PEC byte. Omitted: none.
STOP	CPD		End the transaction with STOP after this query; a following I2C command of the same program message starts a new transaction. Omitted: STOP follows at the end of the program message.

Returns

<Block> Definite-length block #<n><byte count><bytes> with the received bytes in bus order: <count> bytes, or with AUTO the count byte and the counted bytes, followed by the <extra> bytes. Every byte value from 0 to 255 is transported unchanged.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	I2C_SDA or I2C_SCL is not routed. No response is returned.
-222	"Data out of range"	<address> is outside 0 to 127, <count> is outside 1 to 32768, or <extra> is outside 0 to 255. No response is returned.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:I2C2:READ?. No response is returned.
+300	"I2C address not acknowledged"	No device acknowledged the address. The transaction is ended with STOP; no response is returned.
+302	"I2C bus error"	Arbitration was lost, the bus is stuck, or the device stretched the clock beyond the bus timeout of 25 ms. The instrument releases the bus; no response is returned.

Example

```
DIG:I2C:WRIT 80,0,16;READ? 80,4
-> #14<4 bytes>
DIG:I2C:WRIT 11,#H20;READ? 11,AUTO,1
-> #16<count byte 4, 4 data bytes, PEC byte>
```

Related commands

`DIGital:I2C:WRITe`, `DIGital:I2C:FREQuency`, `SYSTem:ERRor[:NEXT]?`

4.10 DIGital:I2C:WRITe

Addresses an I2C device for writing and sends data bytes to it.

The instrument is the bus master and addresses the device with its unshifted 7-bit address: an EEPROM at 0x50 is address `80`, not `160`; 10-bit addressing and slave operation are not supported. Every I2C command is a complete transaction of its own, START, address, data, STOP, but consecutive I2C commands on the same instance within one program message form one transaction: the first sends START, every further one a Repeated START instead of STOP and START, and STOP follows at the end of the program message. `DIG:I2C:WRIT 80,0,16;READ? 80,4` therefore produces START, address 80 write, bytes 0 and 16, Repeated START, address 80 read, 4 bytes, STOP, the random read of an EEPROM; sent as two program messages, the two commands get a STOP between them. STOP is also sent at once by STOP as the last parameter, by an error in the transaction, by Device Clear, by the loss of the connection and when a line of the bus loses its function. Device Clear and the loss of the connection also abort a running command; the bytes already written stay written. After an error the next I2C command of the same program message starts a new transaction, and whether the earlier command failed is visible only in the error queue. Other commands between two I2C commands leave the transaction open and the clock stretched; the pause until the next command is not specified, and the instrument does not guard against the bus timeout of devices such as SMBus devices. A `WRITe` without data bytes addresses the device and ends the transaction, a presence probe. The command completes when the bus activity it causes has ended.

COMMAND

```
DIGital:I2C:WRITe
  <address>[,<byte>[,<byte>...]|<block>][,STOP]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<address>`	NR1	0 to 127	Unshifted 7-bit address of the device.
`<byte>`	NR1	0 to 255	Data byte, decimal or in the non-decimal forms `#H..`, `#Q..` and `#B..`; several bytes are separated by commas and sent in list order. Omitted together with `<block>`: no data, the command probes the address.

<block>	Block	The data bytes as one definite-length block #<n><byte count><bytes> instead of a byte list; the indefinite form #0 is not accepted.
STOP	CPD	End the transaction with STOP after this command; a following I2C command of the same program message starts a new transaction. Omitted: STOP follows at the end of the program message.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	I2C_SDA or I2C_SCL is not routed. Nothing is sent.
-222	"Data out of range"	<address> is outside 0 to 127, or a byte is outside 0 to 255. Nothing is sent.
-223	"Too much data"	More than 32768 data bytes. Nothing is sent.
-108	"Parameter not allowed"	A parameter other than STOP follows the binary block, or a block follows a byte list. Nothing is sent.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:I2C2:WRITe. Nothing is sent.
-161	"Invalid block data"	The block is malformed or truncated, or uses the indefinite-length form. Nothing is sent.
+300	"I2C address not acknowledged"	No device acknowledged the address. The transaction is ended with STOP.
+301	"I2C data not acknowledged"	The device did not acknowledge a data byte. The bytes acknowledged so far have been written; the transaction is ended with STOP.
+302	"I2C bus error"	Arbitration was lost, the bus is stuck, or the device stretched the clock beyond the bus timeout of 25 ms. The instrument releases the bus.

Example

```
DIG:MODE GPIO,GPIO,GPIO,GPIO,I2C_SDA,I2C_SCL
DIG:I2C:WRIT 80
DIG:I2C:WRIT 80,0,16;READ? 80,4
-> #14<4 bytes>
DIG:I2C:WRIT 80,0,16,#H12,#H34
```

Related commands

DIGital:I2C:READ?, DIGital:I2C:FREQuency, DIGital:LINE<n>:MODE, SYSTem:ERRor[:NEXT]?

4.11 DIGital:LINE<n>:MODE

Sets or queries the function to which one line is routed.

Routing is the enable: the function runs from the moment its line is routed and stops when the line is routed back to GPIO. Every value except GPIO may be assigned to one line only, and a line accepts only the values of the line table at the start of this chapter; the new routing is validated against the other five lines like DIGital:MODE, so exchanging two functions between two lines needs a detour through GPIO or the port form. A line that loses a function is released first and the volatile data of the function are discarded: an open I2C transaction ends with STOP, a counter reads 0, a UART that loses its RX line discards its receive buffer and error flags, a PWM stops at its inactive level. A line routed back to GPIO takes its stored state at once. The INHibit function is configured with OUTPut:INHibit:ACTive and OUTPut:INHibit:MODE. The inhibit is level-based: in LIVE and LATChing a line routed to INHibit disables the outputs at once if its input is already active, as an open line is with ACTive HIGH. While no line is routed to INHibit, the input counts as inactive, so routing the line away re-enables the outputs in LIVE and leaves a LATChing latch set until OUTPut:PROTection:CLEar or *RST.

COMMAND

```
DIGital:LINE<n>:MODE
  GPIO|TINPut|TOUTput|FAULt|INHibit|PWM<i>|COUNter|I2C_SDA|I2C_SCL|UART_TX|
  UART_RX|SPI_CLK|SPI_MOSI|SPI_MISO|SPI_CS1|SPI_CS2|SPI_CS3
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
GPIO	CPD		General-purpose open-drain output with the stored state of LINE<n>:STATe. Allowed on any number of lines.
TINPut	CPD		External trigger input, the trigger source EXTernal.
TOUTput	CPD		Trigger output: one pulse per accepted trigger.
FAULt	CPD		Fault output: a level that is asserted while a selected fault condition is present.
INHibit	CPD		External inhibit input that disables every output.
PWM<i>	CPD		Output of PWM instance <i>. Omitted suffix: instance 1.
COUNter	CPD		Input of the pulse counter.
I2C_SDA	CPD		Data line of the I2C master.
I2C_SCL	CPD		Clock line of the I2C master.
UART_TX	CPD		Transmit line of the UART.
UART_RX	CPD		Receive line of the UART.
SPI_CLK	CPD		Clock of the SPI master.
SPI_MOSI	CPD		Data output of the SPI master.
SPI_MISO	CPD		Data input of the SPI master.
SPI_CS1	CPD		Chip select 1 of the SPI master.

SPI_CS2	CPD	Chip select 2 of the SPI master.
SPI_CS3	CPD	Chip select 3 of the SPI master.

QUERY

```
DIGital:LINE<n>:MODE?
```

Returns

<CRD>	The function of the line in short form, for example TOUT; PWM instances with their digit, PWM1, single-instance functions without, COUN, and values with an underscore in full, I2C_SDA.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	The function is already routed to another line. The routing is unchanged.
-224	"Illegal parameter value"	The line cannot provide the function according to the line table, or the value names an instance that does not exist, such as PWM4 or UART2_TX. The routing is unchanged.

Example

```
DIG:LINE6:MODE TOUT
DIG:LINE6:MODE?
-> TOUT
DIG:LINE2:MODE PWM
DIG:LINE2:MODE?
-> PWM1
```

Related commands

DIGital:MODE, DIGital:LINE<n>:STATe, OUTPut:INHibit:MODE, TRIGger[:SEQuence]:SOURce

4.12 DIGital:LINE<n>:READ?

Returns the measured level of one line.

The level is measured in every mode, also on a line driven by PWM, SPI, UART or an external device. A released line reads 0 while an external device drives it Low, so the result differs from LINE<n>:STATe? whenever something else drives a released line. Reading changes nothing.

QUERY

```
DIGital:LINE<n>:READ?
```

Returns

`<NR1>`	`1` when the line is High, `0` when it is Low.

Example

```
DIG:LINE3:STAT?
-> 1
DIG:LINE3:READ?
-> 0
```

Related commands

`DIGital:READ?`, `DIGital:LINE<n>:STATe`, `DIGital:LINE<n>:MODE`

4.13 DIGital:LINE<n>:STATe

Sets or queries the stored output state of one line.

`0` drives the line Low; 1 releases it, and the pull-up takes it High unless an external device drives it Low. The state is stored for every line in every mode: while the line is routed to another function the value is stored without touching the line and becomes effective the moment the line is routed to `GPIO`. The state is the corresponding bit of `DIGital:STATe`. The query returns the stored state, never a measurement.

COMMAND

```
DIGital:LINE<n>:STATe
  0|1
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`0`	NR1		Drive the line Low.
1	NR1		Release the line.

QUERY

```
DIGital:LINE<n>:STATe?
```

Returns

`<NR1>`	1 when the stored state releases the line, `0` when it drives the line Low. Not a measurement.

Example

```
DIG:LINE1:STAT 0
DIG:LINE1:STAT?
-> 0
```

Related commands

`DIGital:STATe`, `DIGital:LINE<n>:READ?`, `DIGital:LINE<n>:MODE`

4.14 DIGital:MODE

Sets or queries the routing of all six lines in one command.

The command takes exactly six mode values, line 1 first, each one a value of `DIGital:LINE<n>:MODE`. The complete target assignment is validated before anything changes: every line must be able to provide its function according to the line table, and every function except GPIO may appear only once; if a value fails, the command generates the corresponding error and no line changes. Lines that lose a function are released first and the volatile data of the function are discarded, then the new functions are configured, and a line routed back to GPIO takes its stored state at once. Because the assignment is validated as a whole, this form exchanges two functions between two lines in one step. The query returns the six functions in the same order.

COMMAND

```
DIGital:MODE
  <m1>,<m2>,<m3>,<m4>,<m5>,<m6>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<m1>	CPD		Function of line 1, any value of DIGital:LINE<n>:MODE.
<m2>	CPD		Function of line 2.
<m3>	CPD		Function of line 3.
<m4>	CPD		Function of line 4.
<m5>	CPD		Function of line 5.
<m6>	CPD		Function of line 6.

QUERY

```
DIGital:MODE?
```

Returns

<CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>

The functions of lines 1 to 6 in short form, PWM instances with their digit, single-instance functions without, and values with an underscore in full, for example TINP, TOUT, FAUL, GPIO, I2C_SDA, I2C_SCL.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A function other than GPIO appears twice. No line changes.
-224	"Illegal parameter value"	A line cannot provide its function according to the line table, or a value names an instance that does not exist, such as PWM4 or UART2_TX. No line changes.
-108	"Parameter not allowed"	More than six values. No line changes.
-109	"Missing parameter"	Fewer than six values. No line changes.

Example

```
DIG:MODE GPIO,TINP,GPIO,GPIO,TOUT,GPIO
DIG:MODE GPIO,TOUT,GPIO,GPIO,TINP,GPIO
DIG:MODE?
-> GPIO,TOUT,GPIO,GPIO,TINP,GPIO
```

Related commands

DIGital:LINE<n>:MODE, DIGital:STATe, DIGital:READ?

4.15 DIGital:PWM<i>:ACTive

Selects or queries the level that the duty cycle of a PWM output refers to.

With HIGH the duty cycle is the fraction of the period in which the line is released, with LOW the fraction in which it is driven Low. The other level is the inactive level, at which a duty cycle of 0 holds the line and at which the output stops when the line loses the PWM function. A change applies at the next period boundary without a glitch.

COMMAND

```
DIGital:PWM<i>:ACTive
  HIGH|LOW
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
HIGH	CPD		The duty cycle is the released, High part of the period.
LOW	CPD		The duty cycle is the driven Low part of the period.

QUERY

```
DIGital:PWM<i>:ACTive?
```

Returns

<CRD> HIGH or LOW.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:PWM4:ACTive. The level is unchanged.

Example

```
DIG:PWM:ACT LOW
DIG:PWM:ACT?
-> LOW
```

Related commands

DIGital:PWM<i>:DCYCle, DIGital:PWM<i>:FREQuency

4.16 DIGital:PWM<i>:DCYCle

Sets or queries the duty cycle of a PWM output.

The duty cycle is the fraction of the period in which the line is at the level selected with PWM<i>:ACTive. 0 holds the line at its inactive level, 100 at its active level. The value is quantized to the timer resolution at the programmed frequency, by the same rule on every PWM instance, and the query returns the duty cycle actually programmed. A change applies at the next period boundary without a glitch.

COMMAND

```
DIGital:PWM<i>:DCYCle
  <percent>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<percent>	NRf, %	0 to 100	Duty cycle in percent of the period.
MINimum	CPD		0 %: the line stays at its inactive level.
MAXimum	CPD		100 %: the line stays at its active level.
DEFault	CPD		Factory default, 50 %.

QUERY

```
DIGital:PWM<i>:DCYCle?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the minimum, 0 %, instead of the setting.
MAXimum	CPD		Return the maximum, 100 %, instead of the setting.
DEFault	CPD		Return the factory default, 50 %, instead of the setting.

Returns

<NR3>, % — The duty cycle actually programmed after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<percent> is outside 0 to 100. The duty cycle is unchanged.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:PWM4:DCYCle. The duty cycle is unchanged.

Example

```
DIG:PWM:DCYC 25
DIG:PWM:DCYC?
-> 2.500000000E+01
DIG:PWM:DCYC? DEF
-> 5.000000000E+01
```

Related commands

DIGital:PWM<i>:ACTive, DIGital:PWM<i>:FREQuency, DIGital:LINE<n>:MODE

4.17 DIGital:PWM<i>:FREQuency

Sets or queries the frequency of a PWM output.

A line routed to PWM<i> outputs a periodic signal from the moment it is routed. PWM1 to PWM3 share one frequency range and the same quantization rules for frequency and duty cycle; nothing depends on which instance or line is used. The frequency is quantized to the timer, and the query returns the frequency actually programmed. A change applies at the next period boundary without a glitch. The released phase of the signal is pulled High by the pull-up, so the maximum specified in the data sheet applies to the internal pull-up without external load. There is no phase, dead-time, complementary or synchronized output.

COMMAND

```
DIGital:PWM<i>:FREQuency
  <hertz>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<hertz>	NRf, Hz	Specified in the data sheet	PWM frequency.
MINimum	CPD		Lowest programmable frequency (specified in the data sheet).
MAXimum	CPD		Highest programmable frequency (specified in the data sheet).

DEFault	CPD	Factory default, 1 kHz.

QUERY

```
DIGital:PWM<i>:FREQuency?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the lowest programmable frequency instead of the setting.
MAXimum	CPD		Return the highest programmable frequency instead of the setting.
DEFault	CPD		Return the factory default, 1 kHz, instead of the setting.

Returns

<NR3>, Hz — The frequency actually programmed after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<hertz> is outside the programmable range. The frequency is unchanged.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:PWM4:FREQuency. The frequency is unchanged.

Example

```
DIG:PWM:FREQ 2 kHz
DIG:PWM:FREQ?
-> 2.000000000E+03
DIG:PWM:FREQ? DEF
-> 1.000000000E+03
```

Related commands

DIGital:PWM<i>:DCYCle, DIGital:PWM<i>:ACTive, DIGital:LINE<n>:MODE

4.18 DIGital:READ?

Returns the measured level of all six lines as a bitfield.

The level of every line is measured in every mode, also on lines driven by PWM, SPI, UART or an external device, with the bit assignment of `DIGital:STATe`: bit 0 is line 1, a 1 is a High level. A released line reads `0` while an external device drives it Low, so the result differs from `DIGital:STATe?` whenever something else drives a released line. Reading changes nothing.

QUERY

```
DIGital:READ?
```

Returns

`<NR1>`	The measured levels of lines 1 to 6 as a bitfield from `0` to 63, 1 = High.

Example

```
DIG:STAT?
-> 63
DIG:READ?
-> 61
```

Related commands

`DIGital:LINE<n>:READ?`, `DIGital:STATe`, `DIGital:MODE`

4.19 DIGital:SPI:CPHase

Sets or queries the clock phase of an SPI master.

The phase selects the clock edge on which data are sampled: with `0` the first clock edge of a bit, with 1 the second. Together with `CPOLarity` it selects the SPI mode.

COMMAND

```
DIGital:SPI:CPHase
  0|1
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
0	NR1		Data are sampled on the first clock edge of a bit.
1	NR1		Data are sampled on the second clock edge of a bit.

QUERY

```
DIGital:SPI:CPHase?
```

Returns

<NR1> 0 or 1.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:CPHase. The phase is unchanged.

Example

```
DIG:SPI:CPH 1
DIG:SPI:CPH?
-> 1
```

Related commands

DIGital:SPI:CPOLarity, DIGital:SPI:ORDer, DIGital:SPI:FREQuency

4.20 DIGital:SPI:CPOLarity

Sets or queries the idle level of the clock of an SPI master.

CPOLarity and CPHase have the usual SPI meaning and together select the SPI mode. With 0 the clock rests Low between transfers, with 1 it rests High.

COMMAND

```
DIGital:SPI:CPOLarity
  0|1
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
0	NR1		The clock rests Low.
1	NR1		The clock rests High.

QUERY

```
DIGital:SPI:CPOLarity?
```

Returns

<NR1> 0 or 1.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:CPOLarity. The polarity is unchanged.

Example

```
DIG:SPI:CPOL 1
DIG:SPI:CPOL?
-> 1
```

Related commands

DIGital:SPI:CPHase, DIGital:SPI:ORDer, DIGital:SPI:FREQuency

4.21 DIGital:SPI:CS<c>:ACTive

Selects or queries the level at which a chip select of an SPI master is asserted.

<c> is the chip select 1 to 3, routed as SPI_CS1 to SPI_CS3. A chip select named in SPI:WRITe or SPI:TRANsfer? is asserted before the first clock edge and deasserted after the last. LOW is the level the instrument drives; with HIGH the asserted level is produced by the pull-up.

COMMAND

```
DIGital:SPI:CS<c>:ACTive
  HIGH|LOW
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
HIGH	CPD		Asserted High: the line is released during the transfer.
LOW	CPD		Asserted Low: the line is driven Low during the transfer.

QUERY

```
DIGital:SPI:CS<c>:ACTive?
```

Returns

<CRD> HIGH or LOW.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:CS1:ACTive. The level is unchanged.

Example

```
DIG:SPI:CS2:ACT HIGH
DIG:SPI:CS2:ACT?
-> HIGH
```

Related commands

DIGital:SPI:WRITe, DIGital:SPI:TRANsfer?, DIGital:LINE<n>:MODE

4.22 DIGital:SPI:FREQuency

Sets or queries the clock frequency of an SPI master.

SPI is an SPI master with a fixed word width of 8 bits; 3-wire, dual, quad and octal modes and slave operation are not offered. The clock is quantized, and the query returns the frequency actually programmed. The rising edges of the clock are produced by the pull-up, so the maximum specified in the data sheet applies to the internal pull-up without external load.

COMMAND

```
DIGital:SPI:FREQuency
  <hertz>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<hertz>	NRf, Hz	Specified in the data sheet	Clock frequency.
MINimum	CPD		Lowest programmable frequency (specified in the data sheet).
MAXimum	CPD		Highest programmable frequency (specified in the data sheet).
DEFault	CPD		Factory default, 100 kHz.

QUERY

```
DIGital:SPI:FREQuency?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the lowest programmable frequency instead of the setting.
MAXimum	CPD		Return the highest programmable frequency instead of the setting.
DEFault	CPD		Return the factory default, 100 kHz, instead of the setting.

Returns

<NR3>, Hz — The frequency actually programmed after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<hertz> is outside the programmable range. The frequency is unchanged.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:FREQuency. The frequency is unchanged.

Example

```
DIG:SPI:FREQ 50 kHz
DIG:SPI:FREQ?
-> 5.000000000E+04
DIG:SPI:FREQ? DEF
-> 1.000000000E+05
```

Related commands

DIGital:SPI:TRANsfer?, DIGital:SPI:CPOLarity, DIGital:SPI:CPHase

4.23 DIGital:SPI:ORDer

Selects or queries the bit order of an SPI master.

Every byte is shifted out and in with the most significant or the least significant bit first. The word width is fixed at 8 bits.

COMMAND

```
DIGital:SPI:ORDer
  MSB|LSB
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MSB	CPD		Most significant bit first.
LSB	CPD		Least significant bit first.

QUERY

```
DIGital:SPI:ORDer?
```

Returns

<CRD> MSB or LSB.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:ORDer. The bit order is unchanged.

Example

```
DIG:SPI:ORD LSB
DIG:SPI:ORD?
-> LSB
```

Related commands

`DIGital:SPI:CPOLarity`, `DIGital:SPI:CPHase`, `DIGital:SPI:TRANsfer?`

4.24 DIGital:SPI:TRANsfer?

Shifts bytes out on an SPI bus and returns the bytes received at the same time.

The bytes are sent with the chip-select handling of `SPI:WRITe`, and the bytes received during the transfer are returned as one block of exactly the same length. Reading without meaningful transmit data is a transfer with dummy bytes, for example `DIG:SPI:TRAN? CS1,#H9F,0,0,0` to read a JEDEC ID; there is no separate read command because SPI always clocks. The query needs `SPI_CLK`, `SPI_MOSI`, `SPI_MISO` and the named chip select routed and completes when the transfer has ended. Device Clear and the loss of the connection abort a running transfer at once and deassert the chip select; the bytes already shifted out stay sent, and no response is returned. The response is always a definite-length block, whatever `FORMat[:DATA]` and `FORMat:BORDer` are set to.

QUERY

```
DIGital:SPI:TRANsfer?
  NONE|CS1|CS2|CS3,<byte>[,<byte>...]|<block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
NONE	CPD		No chip select is driven.
CS1	CPD		Chip select 1 is asserted for the complete transfer.
CS2	CPD		Chip select 2 is asserted for the complete transfer.
CS3	CPD		Chip select 3 is asserted for the complete transfer.
<byte>	NR1	0 to 255	Byte to send, decimal or in the non-decimal forms #H.., #Q.. and #B..; several bytes are separated by commas and sent in list order.
<block>	Block		The bytes as one definite-length block #<n><byte count><bytes> instead of a byte list; the indefinite form #0 is not accepted.

Returns

`<Block>`	Definite-length block `#<n><byte count><bytes>` with one received byte per sent byte, in transfer order. Every byte value from 0 to 255 is transported unchanged.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	SPI_CLK, SPI_MOSI or SPI_MISO is not routed, or the named chip select is not routed. Nothing is sent and no response is returned.
-222	"Data out of range"	A byte is outside 0 to 255, or the byte list or block is empty. Nothing is sent and no response is returned.
-223	"Too much data"	More than 32768 bytes. Nothing is sent and no response is returned.
-108	"Parameter not allowed"	A parameter follows the binary block, or a block follows a byte list. Nothing is sent and no response is returned.
-109	"Missing parameter"	The chip-select selector is missing. Nothing is sent and no response is returned.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:TRANsfer?. No response is returned.
-161	"Invalid block data"	The block is malformed or truncated, or uses the indefinite-length form. Nothing is sent and no response is returned.

Example

```
DIG:SPI:TRAN? CS1,#H9F,0,0,0
-> #14<dummy byte, 3 ID bytes>
DIG:SPI:TRAN? NONE,#12<2 bytes>
-> #12<2 bytes>
```

Related commands

DIGital:SPI:WRITe, DIGital:SPI:CS<c>:ACTive, DIGital:SPI:CPHase, DIGital:LINE<n>:MODE

4.25 DIGital:SPI:WRITe

Shifts bytes out on an SPI bus and discards the received bytes.

The first parameter selects the chip select: with CS1, CS2 or CS3 that chip select is asserted before the first clock edge and deasserted after the last, for the complete command; with NONE no chip select is driven. A chip select cannot be held across two commands. At least one byte is required. The command needs SPI_CLK, SPI_MOSI and the named chip select routed and completes when the transfer has ended. Device Clear and the loss of the connection abort a running transfer at once and deassert the chip select; the bytes already shifted out stay sent.

COMMAND

```
DIGital:SPI:WRITe
  NONE|CS1|CS2|CS3,<byte>[,<byte>...]|<block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
NONE	CPD		No chip select is driven.
CS1	CPD		Chip select 1 is asserted for the complete command.
CS2	CPD		Chip select 2 is asserted for the complete command.
CS3	CPD		Chip select 3 is asserted for the complete command.
<byte>	NR1	0 to 255	Byte to send, decimal or in the non-decimal forms #H.., #Q.. and #B..; several bytes are separated by commas and sent in list order.
<block>	Block		The bytes as one definite-length block #<n><byte count><bytes> instead of a byte list; the indefinite form #0 is not accepted.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	SPI_CLK or SPI_MOSI is not routed, or the named chip select is not routed. Nothing is sent.
-222	"Data out of range"	A byte is outside 0 to 255, or the byte list or block is empty. Nothing is sent.
-223	"Too much data"	More than 32768 bytes. Nothing is sent.
-108	"Parameter not allowed"	A parameter follows the binary block, or a block follows a byte list. Nothing is sent.
-109	"Missing parameter"	The chip-select selector is missing. Nothing is sent.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:SPI2:WRITe. Nothing is sent.
-161	"Invalid block data"	The block is malformed or truncated, or uses the indefinite-length form. Nothing is sent.

Example

```
DIG:MODE SPI_CS1,GPIO,SPI_CLK,SPI_MOSI,SPI_MISO,GPIO
DIG:SPI:WRIT CS1,#H06
DIG:SPI:WRIT CS1,#H02,0,0,#H10,#HAA,#H55
```

Related commands

DIGital:SPI:TRANsfer?, DIGital:SPI:CS<c>:ACTive, DIGital:SPI:FREQuency, DIGital:LINE<n>:MODE

4.26 DIGital:STATe

Sets or queries the stored output state of all six lines as a bitfield.

Bit 0 is line 1 and bit 5 is line 6; a 1 releases the line and a 0 drives it Low. The stored state exists for every line in every mode: setting it while a line is routed to another function stores the value without touching the line, and the value becomes effective the moment the line is routed to GPIO. The query returns the stored states, never a measurement; the measured levels are returned by DIGital:READ?.

COMMAND

```
DIGital:STATe
  <bitfield>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<bitfield>	NR1	0 to 63	Stored states of lines 1 to 6, bit 0 = line 1. 63 releases every line.

QUERY

```
DIGital:STATe?
```

Returns

<NR1> The stored states of lines 1 to 6 as a bitfield from 0 to 63. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<bitfield> is outside 0 to 63. The stored states are unchanged.

Example

```
DIG:STAT 62
DIG:STAT?
-> 62
```

Related commands

DIGital:LINE<n>:STATe, DIGital:READ?, DIGital:MODE

4.27 DIGital:TINPut:SLOPe

Selects or queries the edge of the trigger input that produces the external trigger event.

The line routed to TINPut is the physical trigger input of the trigger model. The selected edge produces one EXTernal event, the opposite edge produces nothing; NEGative is the edge that the open-drain output of another instrument drives. The event is accepted only by an armed operation whose trigger source is EXTernal, and only the accepted event defines t0; an edge that arrives while nothing is armed or while another source is selected is ignored. INITiate with source EXTernal arms the operation also while no line is routed to TINPut; no edge comes then, and the operation waits until ABORt, like BUS without *TRG. There is no programmable minimum pulse width, debounce, qualification or threshold; the input threshold and the minimum recognized pulse width are hardware specifications, and the delay from the selected edge to t0 is a product specification of the data sheet. The slope and the routing of the TINPut line are not frozen by an armed or active operation: a new slope applies from the next edge, and routing the line away stops the EXTernal events, so the operation waits until ABORt.

COMMAND

```
DIGital:TINPut:SLOPe
  POSitive|NEGative
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
POSitive	CPD		Rising edge.
NEGative	CPD		Falling edge.

QUERY

```
DIGital:TINPut:SLOPe?
```

Returns

<CRD>	POS or NEG.

Example

```
DIG:LINE1:MODE TINP
DIG:TINP:SLOP POS
DIG:TINP:SLOP?
-> POS
TRIG:SOUR EXT
INIT ACQ
```

Related commands

`DIGital:LINE<n>:MODE`, `TRIGger[:SEQuence]:SOURce`, `INITiate`

4.28 DIGital:TOUTput:POLarity

Selects or queries the polarity of the trigger output pulse.

The line routed to `TOUTput` emits one pulse of `TOUTput:PWIDth` for every accepted trigger of the triggered operation; the pulse marks t0, whatever the trigger source, `IMMediate` included, and the delay from the accepted trigger to the start of the pulse is a product specification of the data sheet. Events that are not accepted produce no pulse, and there is neither a source selection nor an enable: routing the line is the enable, routing it back to `GPIO` stops the pulses. `POSitive` rests Low and pulses High, `NEGative` rests High and pulses Low; with `NEGative` the leading edge of the pulse is driven by the instrument, with `POSitive` it is produced by the pull-up against the cable and load capacitance. The polarity is not frozen by an armed or active operation; a change applies to the next pulse.

COMMAND

```
DIGital:TOUTput:POLarity
  POSitive|NEGative
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`POSitive`	CPD		Rests Low, pulses High.
`NEGative`	CPD		Rests High, pulses Low.

QUERY

```
DIGital:TOUTput:POLarity?
```

Returns

<CRD>	POS or NEG.

Example

```
DIG:LINE2:MODE TOUT
DIG:TOUT:POL POS
DIG:TOUT:POL?
-> POS
```

Related commands

DIGital:TOUTput:PWIDth, DIGital:LINE<n>:MODE, TRIGger[:SEQuence]:SOURce

4.29 DIGital:TOUTput:PWIDth

Sets or queries the width of the trigger output pulse.

Every accepted trigger produces one pulse of this width on the line routed to TOUTput. The requested width is quantized to the grid of the pulse timer, an exact halfway case to the larger width, and the query returns the width actually programmed. A trigger that is accepted while a pulse is still active does not retrigger, extend or restart it; the pulse is not repeated, so the width stays deterministic and the output never turns into a level. The limits are specified in the data sheet for the internal pull-up without external load; range and grid are the same on every line that can carry TOUTput, so nothing depends on which line is routed. The width is not frozen by an armed or active operation.

COMMAND

```
DIGital:TOUTput:PWIDth
  <seconds>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	Specified in the data sheet	Pulse width.
MINimum	CPD		Shortest programmable width (specified in the data sheet).
MAXimum	CPD		Longest programmable width (specified in the data sheet).
DEFault	CPD		Factory default, 10 us.

QUERY

```
DIGital:TOUTput:PWIDth?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the shortest programmable width instead of the setting.
MAXimum	CPD		Return the longest programmable width instead of the setting.
DEFault	CPD		Return the factory default, 10 us, instead of the setting.

Returns

<NR3>, s — The programmed width after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<seconds> is outside the programmable range. The width is unchanged.

Example

```
DIG:TOUT:PWID 100 us
DIG:TOUT:PWID?
-> 1.000000000E-04
DIG:TOUT:PWID? DEF
-> 1.000000000E-05
```

Related commands

DIGital:TOUTput:POLarity, DIGital:LINE<n>:MODE, TRIGger[:SEQuence]:SOURce

4.30 DIGital:UART:AVAilable?

Returns the number of bytes in the receive buffer of a UART.

The count is the number of received bytes that are buffered at this moment and not yet read with UART:READ?. Reading the count removes nothing.

QUERY

```
DIGital:UART:AVAilable?
```

Returns

<NR1> The number of buffered bytes.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	UART_RX is not routed. No response is returned.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:AVAilable?. No response is returned.

Example

```
DIG:UART:AVA?
-> 5
DIG:UART:READ?
-> #15HELLO
DIG:UART:AVA?
-> 0
```

Related commands

DIGital:UART:READ?, DIGital:UART:CLEar

4.31 DIGital:UART:BAUD

Sets or queries the baud rate of a UART.

UART is an asynchronous serial port at the logic levels of the connector, not an RS-232 level interface; there is no RTS/CTS or XON/XOFF flow control, no break and no 9-bit mode. Any value in the range is accepted and quantized to the baud-rate generator, and the query returns the baud rate actually programmed. The rate applies to transmission and reception.

COMMAND

```
DIGital:UART:BAUD
  <baud>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<baud>	NRf, bit/s	Specified in the data sheet	Baud rate.
MINimum	CPD		Lowest programmable baud rate (specified in the data sheet).
MAXimum	CPD		Highest programmable baud rate (specified in the data sheet).
DEFault	CPD		Factory default, 9600 bit/s.

QUERY

```
DIGital:UART:BAUD?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the lowest programmable baud rate instead of the setting.
MAXimum	CPD		Return the highest programmable baud rate instead of the setting.
DEFault	CPD		Return the factory default, 9600 bit/s, instead of the setting.

Returns

<NR3>, bit/s — The baud rate actually programmed after quantization, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<baud> is outside the programmable range. The baud rate is unchanged.
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:BAUD. The baud rate is unchanged.

Example

```
DIG:UART:BAUD 115200
DIG:UART:BAUD?
-> 1.152000000E+05
DIG:UART:BAUD? DEF
-> 9.600000000E+03
```

Related commands

DIGital:UART:BITS, DIGital:UART:PARity, DIGital:UART:SBITs

4.32 DIGital:UART:BITS

Sets or queries the number of data bits of a UART.

A frame consists of a start bit, the data bits, the parity bit selected with UART:PARity and the stop bits of UART:SBITs. The setting applies to transmission and reception.

COMMAND

```
DIGital:UART:BITS
  7|8
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
7	NR1		Seven data bits.
8	NR1		Eight data bits.

QUERY

```
DIGital:UART:BITS?
```

Returns

<NR1> 7 or 8.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:BITS. The setting is unchanged.

Example

```
DIG:UART:BITS 7
DIG:UART:BITS?
-> 7
```

Related commands

DIGital:UART:PARity, DIGital:UART:SBITs, DIGital:UART:BAUD

4.33 DIGital:UART:CLEar

Discards the bytes in the receive buffer of a UART.

Only the buffered bytes are discarded; the error flags of UART:ERRor?, the settings and the routing are unchanged, and reception continues.

COMMAND

```
DIGital:UART:CLEar
```

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:CLEar. Nothing is discarded.

Example

```
DIG:UART:AVA?
-> 12
DIG:UART:CLE
DIG:UART:AVA?
-> 0
```

Related commands

DIGital:UART:AVAilable?, DIGital:UART:READ?, DIGital:UART:ERRor:CLEar

4.34 DIGital:UART:ERRor:CLEar

Clears the reception error flags of a UART.

The flags PARITY, FRAMING and OVERFLOW are cleared and nothing else: the receive buffer, the settings and the routing are unchanged.

COMMAND

```
DIGital:UART:ERRor:CLEar
```

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:ERRor:CLEar. No flag is cleared.

Example

```
DIG:UART:ERR?
-> 2
DIG:UART:ERR:CLE
DIG:UART:ERR?
-> 0
```

Related commands

DIGital:UART:ERRor?, DIGital:UART:CLEar

4.35 DIGital:UART:ERRor?

Returns the sticky reception error flags of a UART.

A flag is set when the event occurs and stays set until UART:ERRor:CLEar or *RST clears it or the line routed to UART_RX loses its function; reading does not clear it. OVERFLOW means that received data were lost: the buffer was full and the newer bytes were dropped, while the buffered bytes are kept; a hardware receive overrun is reported the same way. Reception errors are not queued in the error queue.

QUERY

```
DIGital:UART:ERRor?
```

Returns

<NR1>	Bitfield of the flags: bit 0 PARITY (1), bit 1 FRAMING (2), bit 2 OVERFLOW (4); 0 when none is set.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:ERRor?. No response is returned.

Example

```
DIG:UART:ERR?
-> 4
DIG:UART:ERR:CLE
DIG:UART:ERR?
-> 0
```

Related commands

DIGital:UART:ERRor:CLEar, DIGital:UART:READ?, DIGital:UART:PARity

4.36 DIGital:UART:PARity

Selects or queries the parity of a UART.

With EVEN or ODD a parity bit follows the data bits of every transmitted frame and is checked in every received frame; a received parity error sets the PARITY flag of UART:ERRor?.

COMMAND

```
DIGital:UART:PARity
  NONE|EVEN|ODD
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
NONE	CPD		No parity bit.
EVEN	CPD		Even parity.
ODD	CPD		Odd parity.

QUERY

```
DIGital:UART:PARity?
```

Returns

<CRD>	NONE, EVEN or ODD.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:PARity. The parity is unchanged.

Example

```
DIG:UART:PAR EVEN
DIG:UART:PAR?
-> EVEN
```

Related commands

DIGital:UART:BITS, DIGital:UART:ERRor?, DIGital:UART:SBITs

4.37 DIGital:UART:READ?

Returns and removes the bytes in the receive buffer of a UART.

Reception runs continuously into a receive buffer of 1024 bytes as long as UART_RX is routed; when that line loses its function, the buffered bytes and the error flags are discarded, while losing only the TX line discards nothing. The query returns every byte buffered at that moment and removes exactly those bytes; bytes that arrive while the response is generated remain for the next READ?. An empty buffer is a valid state and returns #10; the query never waits. Reception errors are not queued in the error queue but reported by UART:ERRor?. The response is always a definite-length block, whatever FORMat[:DATA] and FORMat:BORDer are set to.

QUERY

```
DIGital:UART:READ?
```

Returns

`<Block>`	Definite-length block `#<n><byte count><bytes>` with the received bytes in arrival order, `#10` when the buffer is empty. Every byte value from 0 to 255 is transported unchanged.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	UART_RX is not routed. No response is returned.
-113	`"Undefined header"`	The header names an instance that does not exist, such as `DIGital:UART2:READ?`. No response is returned.

Example

```
DIG:UART:AVA?
-> 5
DIG:UART:READ?
-> #15HELLO
DIG:UART:READ?
-> #10
```

Related commands

`DIGital:UART:AVAilable?`, `DIGital:UART:ERRor?`, `DIGital:UART:CLEar`, `DIGital:UART:WRITe`

4.38 DIGital:UART:SBITs

Sets or queries the number of stop bits of a UART.

The stop bits end every transmitted frame. A framing error in a received frame sets the FRAMING flag of `UART:ERRor?`.

COMMAND

```
DIGital:UART:SBITs
  1|2
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
1	NR1		One stop bit.
2	NR1		Two stop bits.

QUERY

```
DIGital:UART:SBITs?
```

Returns

<NR1> 1 or 2.

Errors

CODE	MESSAGE	CONDITION
-113	"Undefined header"	The header names an instance that does not exist, such as DIGital:UART2:SBITs. The setting is unchanged.

Example

```
DIG:UART:SBIT 2
DIG:UART:SBIT?
-> 2
```

Related commands

DIGital:UART:BITS, DIGital:UART:PARity, DIGital:UART:BAUD

4.39 DIGital:UART:WRITe

Transmits bytes on a UART.

The command is sequential: it completes when the last stop bit has been sent, so a following command sees the transmission finished. A WRITe without bytes sends nothing, but like every WRITe it needs UART_TX routed; a UART also works with only its TX line or only its RX line routed. Device Clear and the loss of the connection stop the transmission at once; the bytes already sent stay sent.

COMMAND

```
DIGital:UART:WRITe
  [<byte>[,<byte>...]|<block>]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<byte>`	NR1	0 to 255	Byte to transmit, decimal or in the non-decimal forms `#H..`, `#Q..` and `#B..`; several bytes are separated by commas and sent in list order. Omitted together with `<block>`: nothing is sent.
`<block>`	Block		The bytes as one definite-length block `#<n><byte count><bytes>` instead of a byte list; the indefinite form `#0` is not accepted.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	UART_TX is not routed, also for a WRITe without bytes. Nothing is sent.
-222	`"Data out of range"`	A byte is outside 0 to 255. Nothing is sent.
-223	`"Too much data"`	More than 32768 bytes. Nothing is sent.
-108	`"Parameter not allowed"`	A parameter follows the binary block, or a block follows a byte list. Nothing is sent.
-113	`"Undefined header"`	The header names an instance that does not exist, such as `DIGital:UART2:WRITe`. Nothing is sent.
-161	`"Invalid block data"`	The block is malformed or truncated, or uses the indefinite-length form. Nothing is sent.

Example

```
DIG:MODE UART_TX,UART_RX,GPIO,GPIO,GPIO,GPIO
DIG:UART:WRIT #15HELLO
DIG:UART:WRIT #H0D,#H0A
```

Related commands

`DIGital:UART:READ?`, `DIGital:UART:BAUD`, `DIGital:LINE<n>:MODE`

5

CHAPTER 5

DISPlay commands

This chapter describes the native SCPI interface to the local front panel: the display presentation state, its brightness and a simple informational message box for the controller. It is deliberately small and carries no virtual keys, touch coordinates, page selection or widget control; complete remote operation of the graphical front panel is provided by the HTML5 web interface, so automation scripts stay independent of the GUI layout. `DISPlay:STATe` and `DISPlay:BRIGhtness` are system settings shared with the local GUI and the web interface; the controller-supplied message is transient. The screenshot query is described with the HCOPy subsystem.

IN THIS CHAPTER

5.1 DISPlay:BRIGhtness 72

5.2 DISPlay:STATe 74

5.3 DISPlay[:WINDow]:TEXT:CLEar 75

5.4 DISPlay[:WINDow]:TEXT[:DATA] 75

5.1 DISPlay:BRIGhtness

Sets or queries the brightness of the local display.

The brightness applies while `DISPlay:STATe` is `ON` and is an integer percentage from 1 to 100, where 1 is the minimum visible setting. Setting it while the display is off updates the stored value without switching the display on; that value is applied the next time `DISPlay:STATe ON` is selected, so switching the display off and on again restores the configured brightness. Brightness never implicitly changes `DISPlay:STATe`. The setting is shared with the local GUI and the HTML5 web interface. The query returns the configured percentage, also while the display is off, and has no side effects.

COMMAND

```
DISPlay:BRIGhtness
  <percent>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<percent>	NR1, %	1 to 100	New brightness in percent. 1 is the minimum visible setting, 100 is maximum brightness.
MINimum	CPD		Minimum visible brightness, 1 %.
MAXimum	CPD		Maximum brightness, 100 %.
DEFault	CPD		Factory default, 100 %.

QUERY

```
DISPlay:BRIGhtness?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the minimum visible brightness, 1, instead of the setting.
MAXimum	CPD		Return the maximum brightness, 100, instead of the setting.
DEFault	CPD		Return the factory default, 100, instead of the setting.

Returns

<NR1>, % The configured brightness from 1 to 100, or the selected capability value. Returned unchanged while the display is off.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<percent> is outside 1 to 100. The brightness is unchanged.

Example

```
DISP:BRIG 40
DISP:BRIG?
-> 40
DISP:BRIG? DEF
-> 100
```

Related commands

DISPlay:STATe, DISPlay[:WINDow]:TEXT[:DATA], HCOPy:DATA?

5.2 DISPlay:STATe

Enables or disables the presentation on the local display.

OFF switches the physical display presentation and its backlight off for rack or unattended operation; the instrument, the measurement system, the communication interfaces, the HTML5 web interface and the logical GUI state continue to run unchanged. ON restores the presentation with the configured DISPlay:BRIGhtness, which a blanked display preserves. The state changes no source, output, measurement, acquisition or local/remote state, and it does not disable screenshots: HCOPy:DATA? captures the logical GUI content and stays available while the display is off. The setting is shared with the local GUI and the HTML5 web interface. The query reports the configured state, not whether a particular refresh or backlight edge is active at that instant.

COMMAND

```
DISPlay:STATe
  ON|OFF|1|0
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		Local display presentation is enabled with the configured brightness.
OFF | 0	Boolean		Local display presentation and backlight are switched off; everything else keeps running.

QUERY

```
DISPlay:STATe?
```

Returns

<NR1>	1 when the local display presentation is enabled, otherwise 0.

Example

```
DISP:STAT OFF
DISP:STAT?
-> 0
HCOP:DATA?
-> #549152<49152 bytes of JPEG data>
DISP:STAT ON
```

Related commands

`DISPlay:BRIGhtness`, `HCOPy:DATA?`, `DISPlay[:WINDow]:TEXT[:DATA]`

5.3 DISPlay[:WINDow]:TEXT:CLEar

Clears the controller-supplied message from the display.

The message box is removed and the normal GUI beneath it becomes visible again. A clear without a message present is a harmless no-op and generates no error. `DISPlay:STATe`, the brightness and every instrument configuration are unchanged. After the clear `DISPlay:TEXT?` returns an empty string.

COMMAND

```
DISPlay[:WINDow]:TEXT:CLEar
```

Example

```
DISP:TEXT "Self test running"
DISP:TEXT:CLE
DISP:TEXT?
-> ""
```

Related commands

`DISPlay[:WINDow]:TEXT[:DATA]`, `DISPlay:STATe`, `HCOPy:DATA?`

5.4 DISPlay[:WINDow]:TEXT[:DATA]

Sets or queries the controller-supplied message shown on the display.

The string is shown as a simple informational message box over the normal front-panel GUI, and a new command replaces the message currently shown. The message is display-only: it changes no source, measurement, trigger, status or local/remote state and it is not an input dialog. It does not switch the display on while `DISPlay:STATe` is OFF, but it stays part of the logical GUI state and therefore appears in `HCOPy:DATA?`. The message is transient and is removed by `DISPlay:TEXT:CLEar`; `*CLS` leaves it unchanged. The query returns the configured message, also while the display is off, and an empty string when none is configured.

COMMAND

```
DISPlay[:WINDow]:TEXT[:DATA]
  "<string>"
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<string>	String		Message text in quotes. It replaces the message currently shown.

QUERY

```
DISPlay[:WINDow]:TEXT[:DATA]?
```

Returns

<String> The configured message in quotes, or "" when no message is configured.

Errors

CODE	MESSAGE	CONDITION
-223	"Too much data"	<string> is longer than the model-dependent maximum message length. The display is unchanged.

Example

```
DISP:TEXT "DUT 1 under test"
DISP:TEXT?
-> "DUT 1 under test"
DISP:TEXT:CLE
```

Related commands

DISPlay[:WINDow]:TEXT:CLEar, DISPlay:STATe, HCOPy:DATA?

6

CHAPTER 6

FETCh commands

This chapter describes how acquired data are read back. The array queries read the acquisition record in the internal sample memory that INITiate ACQuire produces: FETCh:ARRay? reads it non-destructively, FETCh:ARRay:NEXT? consumes it block by block while the acquisition is still running, and FETCh:ARRay:POINts? reports its population and capacity. The scalar queries return the last coherent voltage/current point of a channel without starting a new measurement; a fresh point is taken with the MEASure commands. No FETCh command arms, triggers or waits for an acquisition.

IN THIS CHAPTER

6.1 FETCh:ARRay? 77

6.2 FETCh:ARRay:NEXT? 79

6.3 FETCh:ARRay:POINts? 80

6.4 FETCh[:SCALar]:CURRent[:DC]? 81

6.5 FETCh[:SCALar]:POWer? 82

6.6 FETCh[:SCALar]:VOLTage[:DC]? 83

6.1 FETCh:ARRay?

Returns the valid points of the retained acquisition record, or an index range of them, without consuming them.

The record holds the acquisition time points that INITiate ACQuire has produced, in time order; each point is laid out as the SENSe:FUNCtion selection of the record defines, ascending channel, voltage before current, without placeholders, so a single-channel record with the default selection reads V[1],I[1],V[2],I[2],.... Without parameters every valid point is returned, with <start> the points from that 1-based index to the valid end, with <start>,<end> the inclusive range; indices count time points, not transfer values. The query is non-destructive and repeatable, and it never moves the consuming cursor of FETCh:ARRay:NEXT?. A successful INITiate with an ADC participant discards the previous record at once, and the pre-trigger history of an operation that waits for its trigger is not fetchable. During an active acquisition the number of valid points is snapshotted when the query is accepted, later samples are not appended to the response, and a request beyond that snapshot or for points already consumed by NEXT? is rejected immediately; the query never waits for future samples. The representation follows FORMat[:DATA] and FORMat:BORDer: comma-separated ASCII values, or one IEEE 488.2 definite-length block of binary32 or binary64 values; an overrange reading is 9.900000000E+37 or -9.900000000E+37.

QUERY

```
FETCh:ARRay?
  [<start>[,<end>]]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<start>	NR1	1 to the number of valid points	Acquisition index of the first point to return. Omitted: the first valid point.
<end>	NR1	<start> to the number of valid points	Acquisition index of the last point to return, inclusive. Omitted: the last valid point.

Returns

<NR3> list or <Block> — The requested points in time order and record layout, as ASCII values or as a binary block according to FORMat[:DATA]. Never an empty response.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A DLog acquisition is active; the record belongs to the file. No data are returned.
-222	"Data out of range"	<start> is below 1, or <end> is below <start>. No data are returned.
-230	"Data corrupt or stale"	No valid record point exists, the range lies beyond the valid points snapshotted at acceptance, or it addresses points already consumed by FETCh:ARRay:NEXT?. No data are returned and the record is unchanged.

Example

```
SENS:SWE:POIN 4
INIT
*OPC?
-> 1
FETC:ARR?
-> 4.999870000E+00,1.000210000E-01,4.999910000E+00,1.000190000E-01,4.999900000E
   +00,1.000200000E-01,4.999880000E+00,1.000220000E-01
FETC:ARR? 2,3
-> 4.999910000E+00,1.000190000E-01,4.999900000E+00,1.000200000E-01
```

Related commands

FETCh:ARRay:POINts?, FETCh:ARRay:NEXT?, FORMat[:DATA], SENSe:FUNCtion:VOLTage

6.2 FETCh:ARRay:NEXT?

Returns and consumes the oldest ready block of the acquisition record.

The query is the streaming readout of the same record that FETCh:ARRay? reads: it returns the oldest points not yet consumed and advances the one export cursor, which every interface shares, by exactly the number of points returned, and consumed points are released for reuse by the acquisition, which is how a target longer than the local capacity, or an unbounded one, runs without loss. While the acquisition is active a block is exactly 4096 time points; after it has ended, by reaching its target, by ABORt or by an overrun, full blocks are still returned while at least 4096 points remain, and the final 1 to 4095 points form one last remainder block. The query never waits: it succeeds only while ADC_BUFFER_READY in the Measurement status is set, so a controller polls STATus:MEASurement:CONDition? once per block rather than waiting with *OPC?, which returns only when the whole operation has ended. A failure before the response is committed consumes nothing, and an overrun stops the acquisition before unread points are overwritten, sets ACQ_OVERRUN in the Questionable status and leaves everything still buffered drainable. With FORMat ASCii a block of a single-channel V+I record carries 8192 values; at high rates ASCII conversion may not keep up and ends in an overrun, so binary transfer is recommended.

QUERY

```
FETCh:ARRay:NEXT?
```

Returns

<NR3> list or <Block>	One block of 4096 points, or the final shorter remainder, in time order and record layout, as ASCII values or as a binary block according to FORMat[:DATA].

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A DLog acquisition is active; the record belongs to the file. Nothing is consumed and no data are returned.
-230	"Data corrupt or stale"	ADC_BUFFER_READY is not set: the operation still waits for its trigger, fewer than 4096 unread points exist while the acquisition is active, or no unread point exists after it has ended. Nothing is consumed and no data are returned.

Example

```
FORM REAL,32
SENS:SWE:POIN INF
INIT
STAT:MEAS:COND?
-> 6
FETC:ARR:NEXT?
-> #532768<32768 bytes of binary32 data>
```

Related commands

STATus:MEASurement:CONDition?, FETCh:ARRay?, FETCh:ARRay:POINts?, FORMat[:DATA]

6.3 FETCh:ARRay:POINts?

Returns the number of valid, unconsumed points in the acquisition record, or the capacity of the record.

Without a selector the query returns how many acquisition time points are currently valid in the record: the fill level of a retained record, which during consuming streaming decreases with every FETCh:ARRay:NEXT? and increases while the acquisition adds points, and 0 when no record exists, for example after TRACe:CLEar. With MAXimum it returns how many complete time points the sample memory can hold simultaneously under the current configuration, accounting for the SENSe:FUNCtion selection, the internal storage precision, alignment and reserved metadata; the count is in time points, not bytes or values. The capacity does not limit SENSe:SWEep:POINts, which may be larger when the record is drained with NEXT?; comparing the two before INITiate tells whether a complete retained record is guaranteed. Both forms are non-destructive and change neither the record, the cursor nor any setting, and both are plain ASCII regardless of FORMat[:DATA]. The pre-trigger history kept while an operation waits for its trigger is not counted.

QUERY

```
FETCh:ARRay:POINts?
  [MAXimum]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MAXimum	CPD		Return the local record capacity for the current acquisition configuration instead of the current population.

Returns

<NR1> The number of valid, unconsumed acquisition time points, 0 without a record; with MAXimum the number of points that fit in the record simultaneously.

Example

```
FETC:ARR:POIN?
-> 100000
```

Related commands

SENSe:SWEep:POINts, FETCh:ARRay?, FETCh:ARRay:NEXT?, TRACe:CLEar

6.4 FETCh[:SCALar]:CURRent[:DC]?

Returns the current of the last valid scalar result of a channel without taking a new measurement.

The scalar result of a channel is one coherent voltage/current point: the point that the last MEASure query of that channel took or snapshotted, or the last time point produced by the last completed, aborted or overrun INITiate ACQuire acquisition, which replaces the result of every channel atomically when it ends, whether or not its quantities were recorded. The query returns the signed current of that point, positive when current flows out of the instrument into the load and negative when the instrument sinks it; the voltage and power of the same point are read with the two sibling queries, and reading one quantity invalidates none of the others. Nothing is armed, triggered, consumed or waited for: the record, its population and the FETCh:ARRay:NEXT? cursor are unchanged, and the query is repeatable. The result stays valid with the output off and reflects the terminal current. It is not a fresh measurement; MEASure:CURRent? takes one.

QUERY

```
FETCh[:SCALar]:CURRent[:DC]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, A	The signed current of the last valid scalar point of the channel. Not a new measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-230	"Data corrupt or stale"	No valid scalar result exists for a listed channel, because none has been established yet or the underlying result has been invalidated. No response is returned; the query does not wait for a future result. Also after power-on or *RST before any MEASure query or acquisition has produced a result.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:VOLT?
-> 4.999870000E+00
FETC:CURR?
-> 1.000210000E-01
MEAS:VOLT? (@2)
-> 1.199980000E+01
FETC:CURR? (@1:2)
-> 1.000210000E-01,-2.499500000E-01
```

Related commands

`MEASure[:SCALar]:CURRent[:DC]?`, `FETCh[:SCALar]:VOLTage[:DC]?`, `FETCh[:SCALar]:POWer?`, `INITiate`

6.5 FETCh[:SCALar]:POWer?

Returns the power of the last valid scalar result of a channel without taking a new measurement.

The value is the product of the voltage and the signed current of the one coherent point that forms the scalar result of the channel: the point taken or snapshotted by the last `MEASure` query, or the last time point of the last completed, aborted or overrun `INITiate ACQuire` acquisition, which replaces the result of every channel atomically when it ends. Power is positive when the instrument supplies the load and negative when it absorbs power from it; it is never recomputed from later readings, so it agrees exactly with the voltage and current returned by the two sibling queries. When the voltage or the current of that point is an overrange reading, the power is returned as `9.900000000E+37` or `-9.900000000E+37` with the sign of the product, positive when one factor is 0. Nothing is armed, triggered, consumed or waited for: the record and the `FETCh:ARRay:NEXT?` cursor are unchanged and the query is repeatable. The result stays valid with the output off. It is not a fresh measurement; `MEASure:POWer?` takes one.

QUERY

```
FETCh[:SCALar]:POWer?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, W` — Voltage times signed current of the last valid scalar point of the channel. Not a new measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-230	"Data corrupt or stale"	No valid scalar result exists for a listed channel, because none has been established yet or the underlying result has been invalidated. No response is returned; the query does not wait for a future result. Also after power-on or *RST before any MEASure query or acquisition has produced a result.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
INIT
*OPC?
-> 1
FETC:POW?
-> 5.000920000E-01
FETC:POW? (@1:2)
-> 5.000920000E-01,-1.249750000E+00
```

Related commands

MEASure[:SCALar]:POWer?, FETCh[:SCALar]:VOLTage[:DC]?, FETCh[:SCALar]:CURRent[:DC]?, INITiate

6.6 FETCh[:SCALar]:VOLTage[:DC]?

Returns the voltage of the last valid scalar result of a channel without taking a new measurement.

The scalar result of a channel is one coherent voltage/current point: the point that the last MEASure query of that channel took or snapshotted, or the last time point produced by the last completed, aborted or overrun INITiate ACQuire acquisition, which replaces the result of every channel atomically when it ends, whether or not its quantities were recorded. The query returns the voltage of that point; the current and power of the same point are read with the two sibling queries, which is how a coherent pair is obtained after one MEASure:VOLTage?, and reading one quantity invalidates none of the others. Nothing is armed, triggered, consumed or waited for: the record, its population and the FETCh:ARRay:NEXT? cursor are unchanged, and the query is repeatable. The result stays valid with the output off and reflects the terminal voltage. It is not a fresh measurement; MEASure:VOLTage? takes one.

QUERY

```
FETCh[:SCALar]:VOLTage[:DC]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The voltage of the last valid scalar point of the channel. Not a new measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-230	"Data corrupt or stale"	No valid scalar result exists for a listed channel, because none has been established yet or the underlying result has been invalidated. No response is returned; the query does not wait for a future result. Also after power-on or *RST before any MEASure query or acquisition has produced a result.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:CURR?
-> 1.000210000E-01
FETC:VOLT?
-> 4.999870000E+00
MEAS:CURR? (@2)
-> -2.499500000E-01
FETC:VOLT? (@1:2)
-> 4.999870000E+00,1.199980000E+01
```

Related commands

MEASure[:SCALar]:VOLTage[:DC]?, FETCh[:SCALar]:CURRent[:DC]?, FETCh[:SCALar]:POWer?, INITiate

7

CHAPTER 7

FORMat commands

This chapter describes how numerical data leave the instrument: the external representation of format-dependent measurement data, the byte order of binary payloads and the radix of status-register responses. The three settings are independent of each other and are global instrument state, not a per-session copy, so a change made through any interface is immediately visible to every other session. None of them changes the internal storage precision of a sample, and none of them turns an ordinary scalar or configuration response into binary data.

IN THIS CHAPTER

7.1 FORMat:BORDer 85

7.2 FORMat:SREGister 86

7.3 FORMat[:DATA] 88

7.1 FORMat:BORDer

Selects or queries the byte order of binary numerical payloads.

The order applies to every payload defined as border-sensitive: binary REAL,32 and REAL,64 measurement responses, binary DLog sample payloads and binary TRACe:SAVE record exports. NORMal is big-endian, SWAPped is little-endian, which matches the native order of ordinary controllers. A change affects responses generated afterwards, DLog files started afterwards and exports written afterwards; stored samples are neither altered nor reinterpreted, and a running DLog keeps the byte order it snapshotted when INITiate DLOG was accepted. The binary32 input blocks of [SOURce:]ARB:STReam:DATA, DATA:VOLTage and DATA:CURRent are not affected; they are always little-endian. The setting may be changed while FORMat ASCii is selected; it then has no effect on ASCII output.

COMMAND

```
FORMat:BORDer
  NORMal|SWAPped
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
NORMal	CPD		Most significant byte first, big-endian.
SWAPped	CPD		Least significant byte first, little-endian.

QUERY

```
FORMat:BORDer?
```

Returns

<CRD> NORM or SWAP. Reported also while FORMat ASCii is selected.

Example

```
FORM:BORD NORM
FORM:BORD?
-> NORM
```

Related commands

FORMat[:DATA], FETCh:ARRay?, TRACe:SAVE, INITiate

7.2 FORMat:SREGister

Selects or queries the numeric representation of status-register responses.

The radix applies to status-register queries only: *ESE?, *ESR?, *SRE?, *STB? and the CONDition?, EVENt?, ENABle?, PTRansition? and NTRansition? queries of the status groups. ASCii returns an ordinary decimal value, the other three return IEEE 488.2 non-decimal numeric response data with the prefixes #B, #H and #Q; these are numeric response forms, not arbitrary blocks, and are unrelated to FORMat:BORDer. The setting changes presentation only: register contents, enable masks, transition filters, latched events and the status logic itself are untouched. Scalar measurements, configuration queries, acquisition point counts, error strings, *OPC? and numerical bulk transfers keep their normal representation. The setting is independent of FORMat[:DATA] and FORMat:BORDer and is global instrument state.

COMMAND

```
FORMat:SREGister
  ASCii|BINary|HEXadecimal|OCTal
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ASCii	CPD		Decimal NR1 value, for example 55.
BINary	CPD		Binary form with the prefix #B, for example #B110111.
HEXadecimal	CPD		Hexadecimal form with the prefix #H, for example #H37.
OCTal	CPD		Octal form with the prefix #Q, for example #Q67.

QUERY

```
FORMat:SREGister?
```

Returns

<CRD> ASC, BIN, HEX or OCT.

Example

```
FORM:SREG HEX
FORM:SREG?
-> HEX
STAT:QUES:COND?
-> #H37
MEAS:VOLT?
-> 5.000000000E+00
```

Related commands

STATus:QUEStionable:CONDition?, *STB?, *ESR?, FORMat[:DATA]

7.3 FORMat[:DATA]

Selects or queries the external representation of format-dependent measurement data.

The setting applies to the functions whose own specification declares their data format-dependent: FETCh:ARRay?, FETCh:ARRay:NEXT?, INITiate DLOG and TRACe:SAVE. For a bulk query ASCii transfers comma-separated numeric values without whitespace and without unit suffixes, with up to 9 significant digits for a stored binary32 and up to 17 for a binary64 sample, while REAL,32 and REAL,64 transfer an IEEE 488.2 definite-length block #<n><byte count><payload> of IEEE-754 values in the FORMat:BORDer byte order; indefinite #0 blocks are never used, and the byte count covers the payload only. For DLog ASCii selects CSV file output and the binary forms select raw sample payloads after the text header, snapshotted when INITiate DLOG is accepted. The value order inside one acquisition point is the record layout, ascending channel and voltage before current without placeholders, points in time order; an overrange reading is 9.900000000E+37 or -9.900000000E+37 in every representation. The setting does not change the internal sample storage precision, and scalar MEASure? queries, FETCh:ARRay:POINts?, range, status and histogram queries stay ASCII regardless of it.

COMMAND

```
FORMat[:DATA]
  ASCii|REAL,32|REAL,64
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ASCii	CPD		Comma-separated SCPI numeric ASCII values; CSV file output for DLog.
REAL,32	CPD		IEEE-754 binary32 values, four payload bytes per scalar.
REAL,64	CPD		IEEE-754 binary64 values, eight payload bytes per scalar. A stored binary32 sample is widened for transfer without gaining precision.

QUERY

```
FORMat[:DATA]?
```

Returns

<CRD>	ASC, REAL,32 or REAL,64.

Example

```
FORM REAL,32
FORM?
-> REAL,32
FETC:ARR?
-> #216<16 bytes of binary32 data>
FORM ASC
FETC:ARR?
-> 4.999870000E+00,1.000210000E-01,4.999910000E+00,1.000190000E-01
```

Related commands

FORMat:BORDer, FETCh:ARRay?, FETCh:ARRay:NEXT?, TRACe:SAVE

8

CHAPTER 8

HCOPy commands

This chapter describes the screenshot readback of the local front panel. The instrument returns one coherent GUI frame as a JPEG image in a binary block; there is no screenshot format, quality or size setting, and no file is created. The display state, brightness and the controller-supplied message are DISPlay commands.

IN THIS CHAPTER

8.1 HCOPy:DATA? 90

8.1 HCOPy:DATA?

Returns a screenshot of the current display content as a JPEG image.

The instrument captures one coherent completed GUI frame from the logical framebuffer, never a mixture of two frames or a partially rendered one. Because the source is the logical GUI content and not the LCD panel, the screenshot stays available while DISPlay:STATe is OFF, and a message shown with DISPlay:TEXT is part of the image. The image uses the native display resolution and a fixed product-defined JPEG quality; the JPEG header carries the dimensions, so no format, quality or size command exists. The image bytes are an opaque file payload and are therefore unaffected by FORMat[:DATA], FORMat:BORDer and FORMat:SREGister, and the query writes nothing to USB or internal storage. A failed capture or encode never returns a silently truncated JPEG.

QUERY

```
HCOPy:DATA?
```

Returns

<Block>	IEEE 488.2 definite-length block #<n><byte count><jpeg data> holding the complete JPEG/JFIF file, followed by the response-message terminator. <n> is the number of digits of the byte count, and the byte count covers the image bytes only.

Errors

CODE	MESSAGE	CONDITION
-200	"Execution error"	The capture or the JPEG encoding failed. No block is returned.

Example

```
HCOP:DATA?
-> #549152<49152 bytes of JPEG data>
DISP:STAT OFF
HCOP:DATA?
-> #549018<49018 bytes of JPEG data>
```

Related commands

DISPlay:STATe, DISPlay[:WINDow]:TEXT[:DATA], FORMat[:DATA]

9

CHAPTER 9

MEASure commands

This chapter describes the scalar measurement queries and the results of the continuously maintained statistics window: minimum, maximum, mean and peak-to-peak of voltage, current and power, the sample count and the measurement duration, the energy accumulator and the histogram branch. Voltage and current are acquired as one coherent point on every channel, regardless of the SENSe:FUNCtion record selection, so a scalar query returns one quantity of that point without any averaging, aperture or filter, and its partner is read from the same point with the scalar FETCh commands; every other query in this chapter is passive and returns an accumulated result without starting a measurement. The statistics window is cleared by MEASure:STATistic:RESet; *CLS, output transitions and starting, stopping or aborting an acquisition or a DLog leave it untouched. The measurement ranges are configured with the SENSe commands, the triggered acquisition with INITiate.

IN THIS CHAPTER

9.1 MEASure:HISTogram:CURRent:BIN? 93
9.2 MEASure:HISTogram:CURRent:DATA? 94
9.3 MEASure:HISTogram:CURRent:EDGE? 96
9.4 MEASure:HISTogram:CURRent:POINts? 97
9.5 MEASure:HISTogram:POWer:BIN? 98
9.6 MEASure:HISTogram:POWer:DATA? 99
9.7 MEASure:HISTogram:POWer:EDGE? 100
9.8 MEASure:HISTogram:POWer:POINts? 101
9.9 MEASure:HISTogram:VOLTage:BIN? 102
9.10 MEASure:HISTogram:VOLTage:DATA? 103
9.11 MEASure:HISTogram:VOLTage:EDGE? 105
9.12 MEASure:HISTogram:VOLTage:POINts? 106
9.13 MEASure[:SCALar]:CURRent[:DC]? 107
9.14 MEASure[:SCALar]:CURRent[:DC]:AVERage? 108
9.15 MEASure[:SCALar]:CURRent[:DC]:MAXimum? 109
9.16 MEASure[:SCALar]:CURRent[:DC]:MINimum? 110

9.17 MEASure[:SCALar]:CURRent[:DC]:PTPeak? 111

9.18 MEASure[:SCALar]:ENERgy? 112

9.19 MEASure[:SCALar]:ENERgy:RESet 113

9.20 MEASure[:SCALar]:ENERgy:UNIT 114

9.21 MEASure[:SCALar]:POWer? 115

9.22 MEASure[:SCALar]:POWer:AVERage? 117

9.23 MEASure[:SCALar]:POWer:MAXimum? 118

9.24 MEASure[:SCALar]:POWer:MINimum? 119

9.25 MEASure[:SCALar]:POWer:PTPeak? 120

9.26 MEASure[:SCALar]:STATistic:COUNt? 121

9.27 MEASure[:SCALar]:STATistic:DURation? 122

9.28 MEASure[:SCALar]:STATistic:RESet 123

9.29 MEASure[:SCALar]:VOLTage[:DC]? 124

9.30 MEASure[:SCALar]:VOLTage[:DC]:AVERage? 125

9.31 MEASure[:SCALar]:VOLTage[:DC]:MAXimum? 126

9.32 MEASure[:SCALar]:VOLTage[:DC]:MINimum? 127

9.33 MEASure[:SCALar]:VOLTage[:DC]:PTPeak? 128

9.1 MEASure:HISTogram:CURRent:BIN?

Returns the boundaries and the sample count of one bin of the current histogram.

Bin indices are 1-based and run from 1 to the value of HISTogram:CURRent:POINts?; the response is the lower boundary, the upper boundary and the accumulated count of that one bin; the two boundaries are the values EDGE? returns for that bin and the count is the value DATA? returns. The boundaries are physical currents in amperes and belong to the fixed public axis of the channel, which covers the complete supported current measurement span and never follows SENSe:CURRent:RANGe or its autorange state. The count holds only valid numeric samples of the current statistics window; a reading outside the active measurement range sets READING_OVERFLOW and is not binned, so the sum of all bin counts can be smaller than STATistic:COUNt?. The query is a passive view: it starts no measurement and neither consumes nor clears the histogram, which is cleared by MEASure:STATistic:RESet. Its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:CURRent:BIN?
  <index>[,(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<index>`	NR1	1 to the bin count	Number of the bin to read, counted from 1. The upper limit is the value returned by `HISTogram:CURRent:POINts?`.
`<channel>`	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

`<NR3>,<NR3>,<NR1>` Lower boundary and upper boundary of the bin in A, then its accumulated sample count.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<index>` is smaller than 1 or larger than the bin count, or the addressed channel does not exist. No response is returned.
-224	`"Illegal parameter value"`	The channel list expands to more than one channel. No response is returned.
-171	`"Invalid expression"`	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:CURR:POIN?
-> <N>
MEAS:HIST:CURR:BIN? 4
-> <e3>,<e4>,4381
MEAS:HIST:CURR:BIN? 4,(@2)
-> <e3>,<e4>,0
```

Related commands

`MEASure:HISTogram:CURRent:DATA?`, `MEASure:HISTogram:CURRent:EDGE?`, `MEASure:HISTogram:CURRent:POINts?`, `MEASure[:SCALar]:STATistic:RESet`

9.2 MEASure:HISTogram:CURRent:DATA?

Returns every bin count of the current histogram of one channel.

The response holds exactly as many counts as `HISTogram:CURRent:POINts?` reports, in bin order: the first count belongs to the interval between the first and the second boundary of `EDGE?`, the last to the interval between the last two. The counts are non-negative integers over the valid numeric current samples of the statistics window; overrange readings are excluded and reported through `READING_OVERFLOW`, so the sum of the counts can be smaller than `STATistic:COUNt?`. The axis is self-describing and stable: `EDGE?` gives the boundaries these counts refer to, and neither a manual range change nor autoranging moves a boundary underneath counts already accumulated. The query is passive and non-consuming; accumulation continues and the counts are cleared by `MEASure:STATistic:RESet`. Its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:CURRent:DATA?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR1> per bin	One non-negative sample count per bin, comma-separated in bin order; as many values as HISTogram:CURRent:POINts? reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:CURR:POIN?
-> <N>
MEAS:HIST:CURR:DATA?
-> 0,0,12,4381,<...>,0
MEAS:HIST:CURR:DATA? (@2)
-> 0,0,0,0,<...>,0
```

Related commands

MEASure:HISTogram:CURRent:EDGE?, MEASure:HISTogram:CURRent:POINts?, MEASure:HISTogram:CURRent:BIN?, MEASure[:SCALar]:STATistic:COUNt?

9.3 MEASure:HISTogram:CURRent:EDGE?

Returns the bin boundaries of the current histogram of one channel.

The response holds one value more than HISTogram:CURRent:POINts? reports, in ascending order: bin 1 spans the first two boundaries, bin 2 the second and the third, and the last bin the last two; which side of an internal boundary a sample lying exactly on it belongs to is a product property, but the assignment is deterministic and consistent. The boundaries are physical currents in amperes and are the public description of the axis, so a program never needs to know how the bins are generated. The axis covers the complete supported current measurement span and is deliberately not a single uniform linear division of it, so that currents near zero keep useful resolution; the concrete distribution is implementation-defined and this query returns the actual boundaries. It does not depend on SENSe:CURRent:RANGe or SENSe:CURRent:RANGe:AUTO, stays unchanged while samples accumulate, and a range change resets neither the histogram nor the statistics window. The query is passive, and its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:CURRent:EDGE?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR3>, A per boundary — The bin boundaries in ascending order; one value more than HISTogram:CURRent:POINts? reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:CURR:POIN?
-> <N>
MEAS:HIST:CURR:EDGE?
-> <e0>,<e1>,<...>,<eN>
MEAS:HIST:CURR:EDGE? (@2)
-> <e0>,<e1>,<...>,<eN>
```

Related commands

`MEASure:HISTogram:CURRent:DATA?`, `MEASure:HISTogram:CURRent:POINts?`, `MEASure:HISTogram:CURRent:BIN?`, `SENSe:CURRent:RANGe`

9.4 MEASure:HISTogram:CURRent:POINts?

Returns the number of bins of the current histogram.

The value is the length of the DATA? response and one less than the length of the EDGE? response, and it is the upper index limit of BIN?; a program reads it first and then interprets the two compound results. The histogram geometry of a quantity is identical on every channel of a model, but this is a simple query and therefore returns one value per listed channel. The bin count is implementation-defined and fixed for a model: it does not change with the selected measurement range, with the autorange state or with a statistics reset; EDGE? returns the actual boundaries. The query is passive and starts no measurement.

QUERY

```
MEASure:HISTogram:CURRent:POINts?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The number of user-visible bins of the current histogram.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:CURR:POIN?
-> <N>
MEAS:HIST:CURR:POIN? (@1:2)
-> <N>,<N>
```

Related commands

`MEASure:HISTogram:CURRent:EDGE?`, `MEASure:HISTogram:CURRent:DATA?`, `MEASure:HISTogram:CURRent:BIN?`

9.5 MEASure:HISTogram:POWer:BIN?

Returns the boundaries and the sample count of one bin of the power histogram.

Bin indices are 1-based and run from 1 to the value of `HISTogram:POWer:POINts?`; the response is the lower boundary, the upper boundary and the accumulated count of that one bin. The boundaries are signed powers in watts on the fixed public axis of the channel, which is symmetric around zero and spans the product of the largest supported voltage and current measurement spans in both directions. Negative bins therefore collect the sink operation of the channel. The count holds only valid numeric samples of the current statistics window; a reading outside the active measurement range sets `READING_OVERFLOW` and is not binned, so the sum of all bin counts can be smaller than `STATistic:COUNt?`. The query is passive, does not consume or clear anything, and addresses exactly one channel because its result is compound.

QUERY

```
MEASure:HISTogram:POWer:BIN?
  <index>[,(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<index>`	NR1	1 to the bin count	Number of the bin to read, counted from 1. The upper limit is the value returned by `HISTogram:POWer:POINts?`.
`<channel>`	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

`<NR3>,<NR3>,<NR1>`	Lower boundary and upper boundary of the bin in W, then its accumulated sample count.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<index>` is smaller than 1 or larger than the bin count, or the addressed channel does not exist. No response is returned.
-224	`"Illegal parameter value"`	The channel list expands to more than one channel. No response is returned.
-171	`"Invalid expression"`	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:POW:POIN?
-> <N>
MEAS:HIST:POW:BIN? 4
-> <e3>,<e4>,2190
MEAS:HIST:POW:BIN? 4,(@2)
-> <e3>,<e4>,0
```

Related commands

`MEASure:HISTogram:POWer:DATA?`, `MEASure:HISTogram:POWer:EDGE?`, `MEASure:HISTogram:POWer:POINts?`, `MEASure[:SCALar]:STATistic:RESet`

9.6 MEASure:HISTogram:POWer:DATA?

Returns every bin count of the power histogram of one channel.

The response holds exactly as many counts as `HISTogram:POWer:POINts?` reports, in bin order: the first count belongs to the interval between the first and the second boundary of `EDGE?`, the last to the interval between the last two. The counts are non-negative integers over the valid numeric power samples of the statistics window; overrange readings are excluded and reported through `READING_OVERFLOW`, so the sum of the counts can be smaller than `STATistic:COUNt?`. Because the axis is symmetric around zero, the distribution shows sourcing and sinking of the channel side by side; `EDGE?` gives the boundaries the counts refer to. The query is passive and non-consuming; the counts are cleared by `MEASure:STATistic:RESet`. Its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:POWer:DATA?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<channel>`	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

`<NR1> per bin`	One non-negative sample count per bin, comma-separated in bin order; as many values as `HISTogram:POWer:POINts?` reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:POW:POIN?
-> <N>
MEAS:HIST:POW:DATA?
-> 0,0,7,2190,<...>,0
MEAS:HIST:POW:DATA? (@2)
-> 0,0,0,0,<...>,0
```

Related commands

MEASure:HISTogram:POWer:EDGE?, MEASure:HISTogram:POWer:POINts?, MEASure:HISTogram:POWer:BIN?, MEASure[:SCALar]:STATistic:COUNt?

9.7 MEASure:HISTogram:POWer:EDGE?

Returns the bin boundaries of the power histogram of one channel.

The response holds one value more than HISTogram:POWer:POINts? reports, in ascending order: bin 1 spans the first two boundaries, and the last bin the last two. The axis is signed and symmetric around zero; its outer boundaries are the product of the largest supported voltage and current measurement spans, negative for absorbed and positive for delivered power. Inside that span the distribution need not be uniformly linear, and finer resolution near zero is permitted; the concrete distribution is implementation-defined and this query returns the actual boundaries. The axis is independent of SENSe:VOLTage:RANGe, SENSe:CURRent:RANGe and their autorange states and stays unchanged while samples accumulate. The query is passive, and its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:POWer:EDGE?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

`<NR3>, W per boundary`	The bin boundaries in ascending order; one value more than `HISTogram:POWer:POINts?` reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:POW:POIN?
-> <N>
MEAS:HIST:POW:EDGE?
-> <e0>,<e1>,<...>,<eN>
MEAS:HIST:POW:EDGE? (@2)
-> <e0>,<e1>,<...>,<eN>
```

Related commands

`MEASure:HISTogram:POWer:DATA?`, `MEASure:HISTogram:POWer:POINts?`, `MEASure:HISTogram:POWer:BIN?`, `MEASure[:SCALar]:POWer?`

9.8 MEASure:HISTogram:POWer:POINts?

Returns the number of bins of the power histogram.

The value is the length of the DATA? response and one less than the length of the EDGE? response, and it is the upper index limit of BIN?. The histogram geometry of a quantity is identical on every channel of a model, but this is a simple query and therefore returns one value per listed channel. The bin count is implementation-defined and fixed for a model: it does not change with the selected measurement ranges, with the autorange state or with a statistics reset; EDGE? returns the actual boundaries. The query is passive and starts no measurement.

QUERY

```
MEASure:HISTogram:POWer:POINts?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The number of user-visible bins of the power histogram.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:POW:POIN?
-> <N>
MEAS:HIST:POW:POIN? (@1:2)
-> <N>,<N>
```

Related commands

MEASure:HISTogram:POWer:EDGE?, MEASure:HISTogram:POWer:DATA?, MEASure:HISTogram:POWer:BIN?

9.9 MEASure:HISTogram:VOLTage:BIN?

Returns the boundaries and the sample count of one bin of the voltage histogram.

Bin indices are 1-based and run from 1 to the value of HISTogram:VOLTage:POINts?; the response is the lower boundary, the upper boundary and the accumulated count of that one bin. The boundaries are physical voltages in volts on the fixed public axis of the channel, which covers the complete supported voltage measurement span and never follows SENSe:VOLTage:RANGe or its autorange state. The count holds only valid numeric samples of the current statistics window; a reading outside the active measurement range sets READING_OVERFLOW and is not binned, so the sum of all bin counts can be smaller than STATistic:COUNt?. The query is passive, does not consume or clear anything, and addresses exactly one channel because its result is compound.

QUERY

```
MEASure:HISTogram:VOLTage:BIN?
  <index>[,(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<index>	NR1	1 to the bin count	Number of the bin to read, counted from 1. The upper limit is the value returned by HISTogram:VOLTage:POINts?.
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR3>,<NR3>,<NR1> Lower boundary and upper boundary of the bin in V, then its accumulated sample count.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<index> is smaller than 1 or larger than the bin count, or the addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:VOLT:POIN?
-> <N>
MEAS:HIST:VOLT:BIN? 3
-> <e2>,<e3>,148230
MEAS:HIST:VOLT:BIN? 3,(@2)
-> <e2>,<e3>,0
```

Related commands

MEASure:HISTogram:VOLTage:DATA?, MEASure:HISTogram:VOLTage:EDGE?, MEASure:HISTogram:VOLTage:POINts?, MEASure[:SCALar]:STATistic:RESet

9.10 MEASure:HISTogram:VOLTage:DATA?

Returns every bin count of the voltage histogram of one channel.

The response holds exactly as many counts as HISTogram:VOLTage:POINts? reports, in bin order: the first count belongs to the interval between the first and the second boundary of EDGE?, the last to the interval between the last two. The counts are non-negative integers over the valid numeric voltage samples of the statistics window; overrange readings are excluded and reported through READING_OVERFLOW, so the sum of the counts can be smaller than STATistic:COUNt?. A regulated channel concentrates its samples in very few bins, which makes the distribution a compact check for setpoint deviations and dropouts. The query is passive and non-consuming; the counts are cleared by MEASure:STATistic:RESet. Its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:VOLTage:DATA?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR1> per bin	One non-negative sample count per bin, comma-separated in bin order; as many values as HISTogram:VOLTage:POINts? reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:VOLT:POIN?
-> <N>
MEAS:HIST:VOLT:DATA?
-> 0,0,148230,3,<...>,0
MEAS:HIST:VOLT:DATA? (@2)
-> 0,0,0,0,<...>,0
```

Related commands

MEASure:HISTogram:VOLTage:EDGE?, MEASure:HISTogram:VOLTage:POINts?, MEASure:HISTogram:VOLTage:BIN?, MEASure[:SCALar]:STATistic:COUNt?

9.11 MEASure:HISTogram:VOLTage:EDGE?

Returns the bin boundaries of the voltage histogram of one channel.

The response holds one value more than HISTogram:VOLTage:POINts? reports, in ascending order: bin 1 spans the first two boundaries, and the last bin the last two. The boundaries are physical voltages in volts and are the public description of the axis. The axis covers the complete supported voltage measurement span; its distribution may be linear or nonlinear and is implementation-defined, and this query returns the actual boundaries. It does not depend on SENSe:VOLTage:RANGe or SENSe:VOLTage:RANGe:AUTO, stays unchanged while samples accumulate, and a range change resets neither the histogram nor the statistics window. The query is passive, and its result is compound, so it addresses exactly one channel.

QUERY

```
MEASure:HISTogram:VOLTage:EDGE?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR3>, V per boundary — The bin boundaries in ascending order; one value more than HISTogram:VOLTage:POINts? reports.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	The addressed channel does not exist. No response is returned.
-224	"Illegal parameter value"	The channel list expands to more than one channel. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:VOLT:POIN?
-> <N>
MEAS:HIST:VOLT:EDGE?
-> <e0>,<e1>,<...>,<eN>
MEAS:HIST:VOLT:EDGE? (@2)
-> <e0>,<e1>,<...>,<eN>
```

Related commands

`MEASure:HISTogram:VOLTage:DATA?`, `MEASure:HISTogram:VOLTage:POINts?`, `MEASure:HISTogram:VOLTage:BIN?`, `SENSe:VOLTage:RANGe`

9.12 MEASure:HISTogram:VOLTage:POINts?

Returns the number of bins of the voltage histogram.

The value is the length of the DATA? response and one less than the length of the EDGE? response, and it is the upper index limit of BIN?. The histogram geometry of a quantity is identical on every channel of a model, but this is a simple query and therefore returns one value per listed channel. The bin count is implementation-defined and fixed for a model: it does not change with the selected measurement range, with the autorange state or with a statistics reset; EDGE? returns the actual boundaries. The query is passive and starts no measurement.

QUERY

```
MEASure:HISTogram:VOLTage:POINts?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The number of user-visible bins of the voltage histogram.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:HIST:VOLT:POIN?
-> <N>
MEAS:HIST:VOLT:POIN? (@1:2)
-> <N>,<N>
```

Related commands

`MEASure:HISTogram:VOLTage:EDGE?`, `MEASure:HISTogram:VOLTage:DATA?`, `MEASure:HISTogram:VOLTage:BIN?`

9.13 MEASure[:SCALar]:CURRent[:DC]?

Measures and returns the current of a channel.

Unlike the statistics queries, this measures: while no triggered operation with an ADC participant is armed or active, it acquires exactly one fresh coherent voltage/current point per addressed channel and returns its current. While an operation with an `ACQuire` or `DLOG` participant is armed, waiting for its trigger or active, the query starts no second acquisition and does not disturb the running path: it atomically snapshots the latest valid coherent live point of that path, and generates `-230` as long as no valid point exists. The point becomes the scalar result of the channel, so `FETCh:VOLTage?` returns the voltage of exactly this point without measuring again; the retained acquisition record and the `FETCh:ARRay:NEXT?` export cursor stay untouched. The reading is signed, positive while current flows out of the instrument into the DUT and negative while the instrument absorbs it, and it stays valid with the output off, where it is the actual terminal current. The active measurement range comes from `SENSe:CURRent:RANGe` and its autorange state, and a reading outside it is returned as `9.900000000E+37` or `-9.900000000E+37` and sets `READING_OVERFLOW`.

QUERY

```
MEASure[:SCALar]:CURRent[:DC]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, A` The measured current of one coherent measurement point per listed channel.

Errors

CODE	MESSAGE	CONDITION
`-222`	`"Data out of range"`	A listed channel does not exist. No response is returned and no measurement is made.
`-230`	`"Data corrupt or stale"`	An operation with an ADC participant is armed or active and no valid coherent live point exists yet. No response is returned.
`-171`	`"Invalid expression"`	A channel appears twice in the channel list. No response is returned and no measurement is made.

Example

```
MEAS:CURR?
-> 1.000000000E-01
FETC:VOLT?
-> 5.000000000E+00
MEAS:CURR? (@1:2)
-> 1.000000000E-01,-2.500000000E-01
```

Related commands

`MEASure[:SCALar]:VOLTage[:DC]?`, `FETCh[:SCALar]:CURRent[:DC]?`, `SENSe:CURRent:RANGe`, `INITiate`

9.14 MEASure[:SCALar]:CURRent[:DC]:AVERage?

Returns the mean current over the accumulated statistics window.

The value is the sample-weighted arithmetic mean of the valid numeric current samples of the current statistics window of the channel and is not time weighted; the time-weighted quantity is `MEASure:ENERgy?`. Every measurement point the channel produces contributes exactly once, independently of why it was produced and of how many consumers used it. An overrange sample counts in `STATistic:COUNt?` and `STATistic:DURation?` but is excluded here. The query is passive: it starts no measurement, produces no sample and changes nothing in the window, which is cleared by `MEASure:STATistic:RESet`. Before the first valid numeric sample of a window the response is `9.910000000E+37` and no error is generated.

QUERY

```
MEASure[:SCALar]:CURRent[:DC]:AVERage?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, A` — The signed arithmetic mean current of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:CURR:AVER?
-> 9.987000000E-02
MEAS:CURR:AVER? (@1:2)
-> 9.987000000E-02,-2.501000000E-01
```

Related commands

MEASure[:SCALar]:CURRent[:DC]:MINimum?, MEASure[:SCALar]:CURRent[:DC]:MAXimum?, MEASure[:SCALar]:STATistic:COUNt?, MEASure[:SCALar]:STATistic:RESet

9.15 MEASure[:SCALar]:CURRent[:DC]:MAXimum?

Returns the largest current observed in the accumulated statistics window.

The extreme value refers to the signed current, so the maximum of a channel that only sinks is the least negative sample, not the largest magnitude. Every produced measurement point is examined exactly once; overrange samples are excluded from the comparison although they count in STATistic:COUNt?, so a value beyond the active measurement range never becomes the maximum. The query is passive: it starts no measurement and does not reset or shorten the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:CURRent[:DC]:MAXimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, A — The largest signed current of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:CURR:MAX?
-> 1.043000000E-01
MEAS:CURR:MAX? (@1:2)
-> 1.043000000E-01,-2.480000000E-01
```

Related commands

MEASure[:SCALar]:CURRent[:DC]:MINimum?, MEASure[:SCALar]:CURRent[:DC]:PTPeak?, MEASure[:SCALar]:CURRent[:DC]:AVERage?, MEASure[:SCALar]:STATistic:RESet

9.16 MEASure[:SCALar]:CURRent[:DC]:MINimum?

Returns the smallest current observed in the accumulated statistics window.

The extreme value refers to the signed current, so a sinking phase produces the minimum even when its magnitude is small. Every produced measurement point is examined exactly once; overrange samples are excluded from the comparison although they count in STATistic:COUNt?, so a value beyond the active measurement range never becomes the minimum. The query is passive: it starts no measurement and does not reset the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:CURRent[:DC]:MINimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, A — The smallest signed current of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:CURR:MIN?
-> 9.510000000E-02
MEAS:CURR:MIN? (@1:2)
-> 9.510000000E-02,-2.530000000E-01
```

Related commands

`MEASure[:SCALar]:CURRent[:DC]:MAXimum?`, `MEASure[:SCALar]:CURRent[:DC]:PTPeak?`, `MEASure[:SCALar]:CURRent[:DC]:AVERage?`, `MEASure[:SCALar]:STATistic:RESet`

9.17 MEASure[:SCALar]:CURRent[:DC]:PTPeak?

Returns the peak-to-peak current of the accumulated statistics window.

The value is the maximum minus the minimum of the signed current over the current statistics window, so it is never negative and is available as soon as both extrema have valid content; no accumulator of its own is maintained for it. Overrange samples are excluded from both extrema and therefore do not widen the span. The query is passive: it starts no measurement and does not reset the window, which is cleared by `MEASure:STATistic:RESet`. Before the first valid numeric sample of a window the response is `9.910000000E+37` and no error is generated.

QUERY

```
MEASure[:SCALar]:CURRent[:DC]:PTPeak?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, A — The difference between the largest and the smallest signed current of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:CURR:PTP?
-> 9.200000000E-03
MEAS:CURR:PTP? (@1:2)
-> 9.200000000E-03,5.000000000E-03
```

Related commands

`MEASure[:SCALar]:CURRent[:DC]:MINimum?`, `MEASure[:SCALar]:CURRent[:DC]:MAXimum?`, `MEASure[:SCALar]:STATistic:RESet`

9.18 MEASure[:SCALar]:ENERgy?

Returns the accumulated energy of a channel.

Energy is the time integral of the signed measured power that the statistics subsystem accumulates for the channel, so it is time weighted and is a different quantity from the sample-weighted `POWer:AVERage?`. It is positive when the instrument has net delivered energy to the DUT and negative when it has net absorbed energy from it; overrange samples are excluded from the integration. The value is returned in the unit that `MEASure:ENERgy:UNIT` selects globally for all channels, while the accumulator itself is per channel. The query is passive: it starts no measurement and does not reset the accumulator, which is zeroed by `MEASure:ENERgy:RESet` and by the complete window reset `MEASure:STATistic:RESet`.

QUERY

```
MEASure[:SCALar]:ENERgy?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, Ws or Wh — The accumulated signed energy in the unit selected with `MEASure:ENERgy:UNIT`. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:ENER?
-> 5.000000000E-01
MEAS:ENER:UNIT WH
MEAS:ENER?
-> 1.388888890E-04
MEAS:ENER? (@1:2)
-> 1.388888890E-04,-2.500000000E-05
```

Related commands

MEASure[:SCALar]:ENERgy:UNIT, MEASure[:SCALar]:ENERgy:RESet, MEASure[:SCALar]:POWer:AVERage?, MEASure[:SCALar]:STATistic:DURation?

9.19 MEASure[:SCALar]:ENERgy:RESet

Clears the accumulated energy of a channel.

Only the energy accumulator of the listed channels is set to zero. Minimum, maximum, mean, peak-to-peak, sample count, duration and all histogram counts of the statistics window keep accumulating unchanged; MEASure:STATistic:RESet is the wider command that starts a new window and clears energy along with everything else. Clearing an accumulator that is already zero is not an error. The command changes no source setting, no measurement-range configuration, no acquisition buffer and no protection state, and it neither starts nor stops a measurement, so energy integration continues from zero with the next sample.

COMMAND

```
MEASure[:SCALar]:ENERgy:RESet
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to clear. Omitted: channel 1. The accumulator of every listed channel is set to zero.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No accumulator is cleared.
-171	"Invalid expression"	A channel appears twice in the channel list. No accumulator is cleared.

Example

```
MEAS:ENER?
-> 5.000000000E-01
MEAS:ENER:RES
MEAS:ENER?
-> 0.000000000E+00
MEAS:STAT:COUN?
-> 148230
MEAS:ENER:RES (@1:2)
```

Related commands

`MEASure[:SCALar]:ENERgy?`, `MEASure[:SCALar]:STATistic:RESet`, `MEASure[:SCALar]:ENERgy:UNIT`

9.20 MEASure[:SCALar]:ENERgy:UNIT

Selects or queries the unit in which accumulated energy is returned.

The setting is global: it applies to the `MEASure:ENERgy?` response of every channel and therefore takes no channel list. `WS` returns watt-seconds, numerically identical to joules, `WH` returns watt-hours. Only the presentation is affected: the accumulated value itself is unchanged, so a change of unit neither resets nor rescales an accumulator. The query returns the selected unit, not an energy value.

COMMAND

```
MEASure[:SCALar]:ENERgy:UNIT
  WS|WH
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
WS	CPD		Return energy in watt-seconds, numerically equal to joules.
WH	CPD		Return energy in watt-hours.

QUERY

```
MEASure[:SCALar]:ENERgy:UNIT?
```

Returns

<CRD>	WS for watt-seconds, WH for watt-hours.

Example

```
MEAS:ENER:UNIT WH
MEAS:ENER:UNIT?
-> WH
MEAS:ENER?
-> 1.388888890E-04
```

Related commands

MEASure[:SCALar]:ENERgy?, MEASure[:SCALar]:ENERgy:RESet

9.21 MEASure[:SCALar]:POWer?

Measures and returns the power of a channel.

Unlike the statistics queries, this measures: it uses exactly one coherent voltage/current point per addressed channel and returns the product of its two values in watts. While no triggered operation with an ADC participant is armed or active, that point is freshly acquired; while an operation with an ACQuire or DLOG participant is armed, waiting for its trigger or active, it is the latest valid coherent live point of the running path, snapshotted without starting or disturbing a second acquisition, and -230 is generated as long as no valid point exists. The underlying point becomes the scalar result of the channel, so FETCh:VOLTage? and FETCh:CURRent? return the two factors of exactly this product; the retained acquisition record and the FETCh:ARRay:NEXT? export cursor stay untouched. The reading is positive while the instrument supplies the DUT and negative while it absorbs power from it, and it stays valid with the output off. When the voltage or the current of the point is outside its active measurement range, the power is returned as 9.900000000E+37 or -9.900000000E+37 with the sign of the product, positive when one factor is 0, and READING_OVERFLOW is set.

QUERY

```
MEASure[:SCALar]:POWer?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W — The product of voltage and current of one coherent measurement point per listed channel.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned and no measurement is made.
-230	"Data corrupt or stale"	An operation with an ADC participant is armed or active and no valid coherent live point exists yet. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned and no measurement is made.

Example

```
MEAS:POW?
-> 5.000000000E-01
FETC:VOLT?
-> 5.000000000E+00
FETC:CURR?
-> 1.000000000E-01
MEAS:POW? (@1:2)
-> 5.000000000E-01,-3.000000000E+00
```

Related commands

MEASure[:SCALar]:VOLTage[:DC]?, MEASure[:SCALar]:CURRent[:DC]?, FETCh[:SCALar]:POWer?, MEASure[:SCALar]:ENERgy?

9.22 MEASure[:SCALar]:POWer:AVERage?

Returns the mean power over the accumulated statistics window.

The value is the sample-weighted arithmetic mean of the per-sample power values of the current statistics window, not the product of the mean voltage and the mean current and not the energy divided by the duration; energy remains the time integral and can differ whenever samples are not equidistant. Every produced measurement point contributes exactly once; a point whose voltage or current is overrange has an overrange power, counts in STATistic:COUNt? and STATistic:DURation? but is excluded from the power statistics and the power histogram. The query is passive: it starts no measurement and does not change the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:POWer:AVERage?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W	The signed arithmetic mean power of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:POW:AVER?
-> 4.993000000E-01
MEAS:POW:AVER? (@1:2)
-> 4.993000000E-01,-3.002000000E+00
```

Related commands

MEASure[:SCALar]:POWer:MINimum?, MEASure[:SCALar]:POWer:MAXimum?, MEASure[:SCALar]:ENERgy?, MEASure[:SCALar]:STATistic:RESet

9.23 MEASure[:SCALar]:POWer:MAXimum?

Returns the largest power observed in the accumulated statistics window.

The extreme value refers to the signed power, so the maximum of a channel that only absorbs is the least negative sample, not the largest magnitude. Every produced measurement point is examined exactly once; overrange samples are excluded from the comparison although they count in STATistic:COUNt?. The query is passive: it starts no measurement and does not reset the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:POWer:MAXimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W	The largest signed power of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:POW:MAX?
-> 5.215000000E-01
MEAS:POW:MAX? (@1:2)
-> 5.215000000E-01,-2.976000000E+00
```

Related commands

MEASure[:SCALar]:POWer:MINimum?, MEASure[:SCALar]:POWer:PTPeak?, MEASure[:SCALar]:POWer:AVERage?, MEASure[:SCALar]:STATistic:RESet

9.24 MEASure[:SCALar]:POWer:MINimum?

Returns the smallest power observed in the accumulated statistics window.

The extreme value refers to the signed power, so any phase in which the instrument absorbed power produces the minimum. Every produced measurement point is examined exactly once; over-range samples are excluded from the comparison although they count in STATistic:COUNt?. The query is passive: it starts no measurement and does not reset the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:POWer:MINimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W — The smallest signed power of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:POW:MIN?
-> 4.755000000E-01
MEAS:POW:MIN? (@1:2)
-> 4.755000000E-01,-3.036000000E+00
```

Related commands

MEASure[:SCALar]:POWer:MAXimum?, MEASure[:SCALar]:POWer:PTPeak?, MEASure[:SCALar]:POWer:AVERage?, MEASure[:SCALar]:STATistic:RESet

9.25 MEASure[:SCALar]:POWer:PTPeak?

Returns the peak-to-peak power of the accumulated statistics window.

The value is the maximum minus the minimum of the signed power over the current statistics window, so it is never negative and is available as soon as both extrema have valid content; no accumulator of its own is maintained for it. On a channel that both sources and sinks the span therefore covers both directions of power flow. Overrange samples are excluded from both extrema and do not widen the span. The query is passive: it starts no measurement and does not reset the window, which is cleared by `MEASure:STATistic:RESet`. Before the first valid numeric sample of a window the response is `9.910000000E+37` and no error is generated.

QUERY

```
MEASure[:SCALar]:POWer:PTPeak?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W	The difference between the largest and the smallest signed power of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:POW:PTP?
-> 4.600000000E-02
MEAS:POW:PTP? (@1:2)
-> 4.600000000E-02,6.000000000E-02
```

Related commands

`MEASure[:SCALar]:POWer:MINimum?`, `MEASure[:SCALar]:POWer:MAXimum?`, `MEASure[:SCALar]:STATistic:RESet`

9.26 MEASure[:SCALar]:STATistic:COUNt?

Returns the number of samples accumulated in the statistics window.

Every measurement point a channel produces contributes exactly once, independently of why it was produced and of how many consumers read it. An overrange sample is counted here although it is excluded from the extrema, the mean, the histogram and the energy, so the sum of the histogram bin counts can be smaller than this value. The count is `0` after `MEASure:STATistic:RESet`. It counts samples, not seconds; the accumulated measurement time is `STATistic:DURation?`. The query is passive and produces no sample of its own.

QUERY

```
MEASure[:SCALar]:STATistic:COUNt?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The number of samples accumulated in the current statistics window, overrange samples included.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:STAT:COUN?
-> 148230
MEAS:STAT:COUN? (@1:2)
-> 148230,148230
```

Related commands

`MEASure[:SCALar]:STATistic:DURation?`, `MEASure[:SCALar]:STATistic:RESet`, `MEASure:HISTogram:CURRent:DATA?`

9.27 MEASure[:SCALar]:STATistic:DURation?

Returns the accumulated measurement duration of the statistics window.

The duration is the measurement time the produced samples represent, not the wall-clock time since the window was started: a period in which the measurement pipeline produces no sample adds nothing merely because real time passes. Overrange samples contribute their time like valid ones, exactly as they contribute to STATistic:COUNt?. The value is 0 after MEASure:STATistic:RESet. Together with the sample count it tells how much measurement the accumulated statistics rest on. The query is passive and starts no measurement.

QUERY

```
MEASure[:SCALar]:STATistic:DURation?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The accumulated measurement duration of the current statistics window.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:STAT:DUR?
-> 1.482300000E+02
MEAS:STAT:DUR? (@1:2)
-> 1.482300000E+02,1.482300000E+02
```

Related commands

MEASure[:SCALar]:STATistic:COUNt?, MEASure[:SCALar]:STATistic:RESet, MEASure[:SCALar]:ENERgy?

9.28 MEASure[:SCALar]:STATistic:RESet

Starts a new statistics window on a channel.

The complete window of every listed channel is cleared in one atomic step: minimum, maximum, mean accumulation, peak-to-peak, sample count, duration, the energy accumulator and every histogram bin count. Histogram geometry is not re-derived from the current measurement-range configuration; the bin boundaries stay based on the maximum supported measurement span and are the same before and after the reset. It is the wider form of `MEASure:ENERgy:RESet`, which clears the energy accumulator alone. The command changes no source setting, no measurement-range configuration, no acquisition buffer and no protection state, and it neither starts nor stops a measurement.

COMMAND

```
MEASure[:SCALar]:STATistic:RESet
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to reset. Omitted: channel 1. A new window is started on every listed channel.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No window is reset.
-171	"Invalid expression"	A channel appears twice in the channel list. No window is reset.

Example

```
MEAS:STAT:RES
MEAS:STAT:COUN?
-> 0
MEAS:ENER?
-> 0.000000000E+00
MEAS:STAT:RES (@1:2)
```

Related commands

`MEASure[:SCALar]:STATistic:COUNt?`, `MEASure[:SCALar]:STATistic:DURation?`, `MEASure[:SCALar]:ENERgy:RESet`, `MEASure:HISTogram:VOLTage:DATA?`

9.29 MEASure[:SCALar]:VOLTage[:DC]?

Measures and returns the voltage of a channel.

Unlike the statistics queries, this measures: while no triggered operation with an ADC participant is armed or active, it acquires exactly one fresh coherent voltage/current point per addressed channel and returns its voltage. While an operation with an ACQuire or DLOG participant is armed, waiting for its trigger or active, the query starts no second acquisition and does not disturb the running path: it atomically snapshots the latest valid coherent live point of that path, and generates -230 as long as no valid point exists. The point becomes the scalar result of the channel, so FETCh:CURRent? returns the current of exactly this point without measuring again; the retained acquisition record and the FETCh:ARRay:NEXT? export cursor stay untouched. The reading stays valid with the output off, where it is the actual terminal voltage, and it is the measured voltage, not the programmed setpoint. The active measurement range comes from SENSe:VOLTage:RANGe and its autorange state, and a reading outside it is returned as 9.900000000E+37 or -9.900000000E+37 and sets READING_OVERFLOW.

QUERY

```
MEASure[:SCALar]:VOLTage[:DC]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The measured voltage of one coherent measurement point per listed channel.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned and no measurement is made.
-230	"Data corrupt or stale"	An operation with an ADC participant is armed or active and no valid coherent live point exists yet. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned and no measurement is made.

Example

```
MEAS:VOLT?
-> 5.000000000E+00
FETC:CURR?
-> 1.000000000E-01
MEAS:VOLT? (@1:2)
-> 5.000000000E+00,1.200000000E+01
```

Related commands

`MEASure[:SCALar]:CURRent[:DC]?`, `FETCh[:SCALar]:VOLTage[:DC]?`, `SENSe:VOLTage:RANGe`, `INITiate`

9.30 MEASure[:SCALar]:VOLTage[:DC]:AVERage?

Returns the mean voltage over the accumulated statistics window.

The value is the sample-weighted arithmetic mean of the valid numeric voltage samples of the current statistics window of the channel; it is not time weighted. Every measurement point the channel produces contributes exactly once, independently of why it was produced and of how many consumers used it. An overrange sample counts in `STATistic:COUNt?` and `STATistic:DURation?` but is excluded here. The query is passive: it starts no measurement, produces no sample and changes nothing in the window, which is cleared by `MEASure:STATistic:RESet`. Before the first valid numeric sample of a window the response is `9.910000000E+37` and no error is generated.

QUERY

```
MEASure[:SCALar]:VOLTage[:DC]:AVERage?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` — The arithmetic mean voltage of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:VOLT:AVER?
-> 4.999300000E+00
MEAS:VOLT:AVER? (@1:2)
-> 4.999300000E+00,1.200010000E+01
```

Related commands

MEASure[:SCALar]:VOLTage[:DC]:MINimum?, MEASure[:SCALar]:VOLTage[:DC]:MAXimum?, MEASure[:SCALar]:STATistic:COUNt?, MEASure[:SCALar]:STATistic:RESet

9.31 MEASure[:SCALar]:VOLTage[:DC]:MAXimum?

Returns the largest voltage observed in the accumulated statistics window.

The value is the largest valid numeric voltage sample of the current statistics window of the channel, which makes it the compact way to catch an overshoot without recording an array. Every produced measurement point is examined exactly once; overrange samples are excluded from the comparison although they count in STATistic:COUNt?, so a value beyond the active measurement range never becomes the maximum. The query is passive: it starts no measurement and does not reset or shorten the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:VOLTage[:DC]:MAXimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The largest voltage of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:VOLT:MAX?
-> 5.012000000E+00
MEAS:VOLT:MAX? (@1:2)
-> 5.012000000E+00,1.201400000E+01
```

Related commands

MEASure[:SCALar]:VOLTage[:DC]:MINimum?, MEASure[:SCALar]:VOLTage[:DC]:PTPeak?, MEASure[:SCALar]:VOLTage[:DC]:AVERage?, MEASure[:SCALar]:STATistic:RESet

9.32 MEASure[:SCALar]:VOLTage[:DC]:MINimum?

Returns the smallest voltage observed in the accumulated statistics window.

The value is the smallest valid numeric voltage sample of the current statistics window of the channel, which makes it the compact way to catch a dropout or a load step without recording an array. Every produced measurement point is examined exactly once; overrange samples are excluded from the comparison although they count in STATistic:COUNt?, so a value beyond the active measurement range never becomes the minimum. The query is passive: it starts no measurement and does not reset the window, which is cleared by MEASure:STATistic:RESet. Before the first valid numeric sample of a window the response is 9.910000000E+37 and no error is generated.

QUERY

```
MEASure[:SCALar]:VOLTage[:DC]:MINimum?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The smallest voltage of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:VOLT:MIN?
-> 4.987000000E+00
MEAS:VOLT:MIN? (@1:2)
-> 4.987000000E+00,1.198600000E+01
```

Related commands

`MEASure[:SCALar]:VOLTage[:DC]:MAXimum?`, `MEASure[:SCALar]:VOLTage[:DC]:PTPeak?`, `MEASure[:SCALar]:VOLTage[:DC]:AVERage?`, `MEASure[:SCALar]:STATistic:RESet`

9.33 MEASure[:SCALar]:VOLTage[:DC]:PTPeak?

Returns the peak-to-peak voltage of the accumulated statistics window.

The value is the maximum minus the minimum voltage over the current statistics window, so it is never negative and is available as soon as both extrema have valid content; no accumulator of its own is maintained for it. It answers how far the regulated output moved during the window, without saying when it moved; that needs an acquisition record. Overrange samples are excluded from both extrema and do not widen the span. The query is passive: it starts no measurement and does not reset the window, which is cleared by `MEASure:STATistic:RESet`. Before the first valid numeric sample of a window the response is `9.910000000E+37` and no error is generated.

QUERY

```
MEASure[:SCALar]:VOLTage[:DC]:PTPeak?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The difference between the largest and the smallest voltage of the statistics window. Not a fresh measurement.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
MEAS:VOLT:PTP?
-> 2.500000000E-02
MEAS:VOLT:PTP? (@1:2)
-> 2.500000000E-02,2.800000000E-02
```

Related commands

MEASure[:SCALar]:VOLTage[:DC]:MINimum?, MEASure[:SCALar]:VOLTage[:DC]:MAXimum?, MEASure[:SCALar]:STATistic:RESet

10

CHAPTER 10

MMEMory commands

This chapter describes the generic file interface of the instrument: listing one directory, querying free space and capacity, transferring file contents in both directions, deleting a file or a directory tree and formatting the external USB stick. Paths are absolute strings in quotes with / as the separator and an optional case-insensitive volume prefix, USB: for the external USB storage, which is also what a path without a prefix means, or INT: for a restricted internal store that the present product does not provide; there is no current directory and no . or .. traversal. A failed command reports its error in the error queue of the session only; it never sets the STORAGE_ERROR status bit, which is reserved for an instrument data path such as DLog.

IN THIS CHAPTER

10.1 MMEMory:CAPacity? 130

10.2 MMEMory:DATA:APPend 132

10.3 MMEMory:DATA[:DEFinite] 133

10.4 MMEMory:DELete 135

10.5 MMEMory:FREE? 136

10.6 MMEMory:INITialize 138

10.7 MMEMory:LIST? 139

10.1 MMEMory:CAPacity?

Returns the total usable capacity of a storage volume in bytes.

The argument names a storage volume in the path notation described at the beginning of this chapter, never a file or a subdirectory; omitted it addresses the external USB storage. The value is the total usable capacity of the mounted filesystem, not the nominal size printed on the medium, and it does not change while files are written. Used space is derived as the capacity minus MMEMory:FREE?; no query returns it directly, and there is no media-present query either, because absent or unusable media are reported through the mass-storage errors. The query alters no file and no allocation state and is permitted while an instrument function writes to the medium.

QUERY

```
MMEMory:CAPacity?
  ["<storage>"]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<storage>`	String		Storage volume to query, in quotes: `USB:` or, where implemented, `INT:`. Omitted: `USB:`. A file or directory path is not a valid argument.

Returns

`<NR1>, bytes` The total usable capacity of the mounted volume as an integer byte count.

Errors

CODE	MESSAGE	CONDITION
-250	`"Mass storage error"`	The volume could not be interrogated because of another filesystem failure.
-251	`"Missing mass storage"`	The named volume is not available on the instrument, for example the reserved `INT:` volume.
-252	`"Missing media"`	No USB mass-storage medium is present.
-253	`"Corrupt media"`	The medium is unusable and could not be mounted.
-257	`"File name error"`	The argument is not a supported storage-volume name.

Example

```
MMEM:CAP?
-> 15678308352
MMEM:CAP? "USB:"
-> 15678308352
```

Related commands

`MMEMory:FREE?`, `MMEMory:LIST?`, `MMEMory:INITialize`

10.2 MMEMory:DATA:APPend

Appends a block of bytes to the end of a file.

The instrument determines the current end of file when the command executes and writes the block there, so the controller neither queries nor tracks the file length; every append is independent and no file position is kept between commands. A file that does not exist is created with the block as its contents, and missing parent directories in the path are created automatically. An empty block is valid: on a new path it creates an empty file, on an existing file it changes nothing. The bytes are stored exactly as received and are opaque to the instrument, so FORMat[:DATA] and FORMat:BORDer never apply to file contents. Together with MMEMory:DATA for the first chunk this is the normal way to upload a large file in bounded pieces; the largest block one command can carry is bounded by the SCPI input buffer (TBD) and never exceeds 999 999 999 bytes.

COMMAND

```
MMEMory:DATA:APPend
  "<path>",<block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<path>	String		File to append to, absolute and in quotes; see the path model at the beginning of this chapter. Created together with its missing parent directories when it does not exist.
<block>	Block		IEEE 488.2 definite-length block #<n><byte count><payload> whose payload is appended unchanged. <n> is the number of digits of the byte count, and the byte count covers the payload only. An empty block is permitted.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An active instrument function owns the target file, the file of a running DLog for example. Nothing is written.
-250	"Mass storage error"	The write failed because of another filesystem failure.
-251	"Missing mass storage"	The path names a storage volume that is not available on the instrument.
-252	"Missing media"	No USB mass-storage medium is present. Nothing is written.
-253	"Corrupt media"	The medium is unusable. Nothing is written.
-254	"Media full"	The free space was insufficient for the complete block. Whether part of the block was written before the failure is not fixed by the spec (TBD).
-257	"File name error"	The path is syntactically invalid or not supported by the storage. Nothing is written.
-258	"Media protected"	The medium or the path is write-protected, for example on a restricted INT: volume. Nothing is written.

Example

```
MMEM:DATA "/dac/run.csv",#3256<256 bytes of CSV data>
MMEM:DATA:APP "/dac/run.csv",#3256<256 bytes of CSV data>
MMEM:DATA:APP "/dac/run.csv",#3128<128 bytes of CSV data>
MMEM:LIST? "/dac"
-> "run.csv",FILE,640
```

Related commands

`MMEMory:DATA[:DEFinite]`, `MMEMory:LIST?`, `MMEMory:DELete`

10.3 MMEMory:DATA[:DEFinite]

Writes a file, or reads it back, as a definite-length binary block.

Without an offset the setting form creates the file, or truncates an existing file to zero and replaces its contents; missing parent directories are created automatically, and an empty block creates or empties a file. With an offset the file is not truncated: the block overwrites the bytes from that zero-based position and the file grows if the block reaches beyond the previous end; only offset zero creates a file that does not exist. The query form returns the complete file, the bytes from an offset to the end of file, or at most a given number of bytes from an offset, which is how a large file is downloaded in bounded chunks. Reading is non-destructive, is not refused while an instrument function writes elsewhere on the medium, and leaves the file unchanged. The bytes are opaque in both directions, `FORMat[:DATA]` and `FORMat:BORDer` never apply to file contents, and both forms are stateless: no handle and no file position remain behind. A write completes only when the file has been flushed and closed.

COMMAND

```
MMEMory:DATA[:DEFinite]
  "<path>"[,<offset>],<block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<path>`	String		File to write, absolute and in quotes; see the path model at the beginning of this chapter. Created together with its missing parent directories when it does not exist.
`<offset>`	NR1, bytes	0 to the current file size	Zero-based position at which the block is written, without truncating the file. Omitted: the file is truncated to zero and completely replaced. Equal to the current size the write is an append; a larger value is rejected.

<block>	Block	IEEE 488.2 definite-length block #<n><byte count><payload> whose payload is written unchanged. <n> is the number of digits of the byte count. An empty block is permitted; the largest block is bounded by the SCPI input buffer (TBD).

QUERY

```
MMEMory:DATA[:DEFinite]?
  "<path>"[,<offset>[,<length>]]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<path>	String		File to read, absolute and in quotes. The file has to exist.
<offset>	NR1, bytes	0 to the file size	Zero-based position of the first returned byte. Omitted: 0. Equal to the file size an empty block is returned; a larger value is an error.
<length>	NR1, bytes	0 to TBD	Maximum number of bytes to return, bounded by the SCPI output buffer. Omitted: everything from the offset to the end of file. A value beyond the remaining bytes returns only those; 0 returns an empty block.

Returns

<Block> IEEE 488.2 definite-length block #<n><byte count><payload> holding the requested file bytes exactly as stored, followed by the response-message terminator. <n> is the number of digits of the byte count, the byte count covers the payload only, and indefinite #0 blocks are never used.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An active instrument function owns the target file, the file of a running DLog for example. Nothing is written; readback of that file is not refused.
-222	"Data out of range"	<offset> is negative or greater than the current file size, so the write would leave a gap or the read would start beyond the end. Nothing is written and no data are returned.
-250	"Mass storage error"	The transfer failed because of another filesystem failure.
-251	"Missing mass storage"	The path names a storage volume that is not available on the instrument.
-252	"Missing media"	No USB mass-storage medium is present. Nothing is written and no data are returned.
-253	"Corrupt media"	The medium is unusable. Nothing is written and no data are returned.

-254	"Media full"	The free space was insufficient for the complete block. Whether part of the block was written before the failure is not fixed by the spec (TBD).
-256	"File name not found"	The file to read does not exist, or a write with a nonzero offset addresses a file that does not exist. Nothing is written and no data are returned.
-257	"File name error"	The path is syntactically invalid, names a directory or is not supported by the storage. Nothing is written and no data are returned.
-258	"Media protected"	The medium or the path is write-protected, for example on a restricted `INT:` volume. Nothing is written.

Example

```
MMEM:DATA "/dac/test.csv",#3120<120 bytes of CSV data>
MMEM:DATA? "/dac/test.csv"
-> #3120<120 bytes of CSV data>
MMEM:DATA "/dac/test.csv",64,#216<16 bytes of CSV data>
MMEM:DATA? "/dac/test.csv",64,16
-> #216<16 bytes of CSV data>
```

Related commands

`MMEMory:DATA:APPend`, `MMEMory:LIST?`, `MMEMory:DELete`, `FORMat[:DATA]`

10.4 MMEMory:DELete

Deletes a file or a complete directory tree.

A path that identifies a regular file deletes that file; a path that identifies a directory deletes its contents recursively and then the directory itself, which is why the command set needs no separate directory-removal command and no recursion option. Deletion is immediate and permanent, there is no confirmation step, and the root of a storage volume is not a valid target. A target that does not exist is an error, not a silent no-op. A file that an active instrument function owns, the file of a running DLog for example, is refused instead of being removed under the writer, and a restricted `INT:` volume may refuse deletion while still allowing readback. Deleting never changes the retained acquisition record or any instrument setting that names the path, such as `[SOURce:]ARB:SEQuence:FILE`.

COMMAND

```
MMEMory:DELete
  "<path>"
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<path>	String		File or directory to delete, absolute and in quotes; see the path model at the beginning of this chapter. A directory is deleted with its complete contents.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An active instrument function owns the target, the file of a running DLog for example. Nothing is deleted.
-250	"Mass storage error"	The deletion failed because of another filesystem failure. Part of a directory tree may already be gone.
-251	"Missing mass storage"	The path names a storage volume that is not available on the instrument.
-252	"Missing media"	No USB mass-storage medium is present. Nothing is deleted.
-253	"Corrupt media"	The medium is unusable. Nothing is deleted.
-256	"File name not found"	The file or directory does not exist. Nothing is deleted.
-257	"File name error"	The path is syntactically invalid, addresses the root of a storage volume or is not supported by the storage. Nothing is deleted.
-258	"Media protected"	The medium or the path is write-protected, for example on a restricted INT: volume. Nothing is deleted.

Example

```
MMEM:DEL "/dac/test.csv"
MMEM:DEL "/logs"
MMEM:LIST?
-> "dac",DIR,0
```

Related commands

MMEMory:LIST?, MMEMory:DATA[:DEFinite], MMEMory:INITialize

10.5 MMEMory:FREE?

Returns the free space of a storage volume in bytes.

The argument names a storage volume in the path notation described at the beginning of this chapter, never a file or a subdirectory; omitted it addresses the external USB storage. The value is the allocation space currently available on the mounted filesystem, so a file smaller than the reported number can still fail to fit once cluster rounding is taken into account. Together with MMEMory:CAPacity? it yields the used space, which no query returns directly. The query alters no file and no allocation state, and it is permitted while an instrument function writes to the medium, where the value it returns is a snapshot that shrinks as the recording grows.

QUERY

```
MMEMory:FREE?
  ["<storage>"]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<storage>`	String		Storage volume to query, in quotes: `USB:` or, where implemented, `INT:`. Omitted: `USB:`. A file or directory path is not a valid argument.

Returns

`<NR1>, bytes` — The currently available free space of the mounted volume as an integer byte count.

Errors

CODE	MESSAGE	CONDITION
-250	`"Mass storage error"`	The volume could not be interrogated because of another filesystem failure.
-251	`"Missing mass storage"`	The named volume is not available on the instrument, for example the reserved `INT:` volume.
-252	`"Missing media"`	No USB mass-storage medium is present.
-253	`"Corrupt media"`	The medium is unusable and could not be mounted.
-257	`"File name error"`	The argument is not a supported storage-volume name.

Example

```
MMEM:FREE?
-> 12683575296
MMEM:FREE? "USB:"
-> 12683575296
```

Related commands

`MMEMory:CAPacity?`, `MMEMory:LIST?`, `MMEMory:INITialize`

10.6 MMEMory:INITialize

Formats the attached external USB mass-storage device.

The command takes no parameter and always targets the external USB stick on the host port; it can never format INT:, which stays outside its scope even though the same path model exposes it. All files, directories and filesystem metadata on the stick are destroyed, and no confirmation query or second command exists: sending MMEMory:INITialize is the destructive request. The medium is formatted as FAT12, FAT16 or FAT32, selected automatically from the resulting cluster count, so no filesystem-type parameter is offered. A stick whose filesystem is unsupported or corrupt can be formatted as well, provided the block device itself is present and writable; a successful mount of the old filesystem is not a prerequisite. The command completes when formatting has finished and the new filesystem is mounted and ready for normal file access; it is refused while DLog or another instrument function owns the storage, and *RST, *CLS, *SAV and *RCL never format storage.

COMMAND

```
MMEMory:INITialize
```

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	DLog or another instrument function owns the USB storage. Nothing is formatted and the medium keeps its contents.
-250	"Mass storage error"	Formatting failed because of another storage failure. The state of the medium is undefined.
-252	"Missing media"	No USB mass-storage medium is present. Nothing is formatted.
-258	"Media protected"	The medium is write-protected. Nothing is formatted.

Example

```
MMEM:INIT
*OPC?
-> 1
MMEM:LIST?
-> ""
```

Related commands

MMEMory:FREE?, MMEMory:CAPacity?, MMEMory:DELete

10.7 MMEMory:LIST?

Returns the entries immediately contained in one directory.

The query lists files and subdirectories of the named directory and is not recursive; `.` and `..` are never part of the result, and implementation files that are deliberately not user-visible may be omitted, particularly on a restricted `INT:` volume. Omitting the argument lists the root of the external USB storage; the path otherwise has to identify an existing directory, and a valid path that names a file is refused rather than listed as a single entry. The query creates, opens, modifies and deletes nothing, and `FORMat[:DATA]` and `FORMat:BORDer` do not apply, because the response is metadata and not numeric bulk data. The directory is read page by page while the response is already going out, so the response deliberately carries no entry count; the triplets are self-delimiting and the names are string data, which keeps the sequence unambiguous. The order of the entries is not defined.

QUERY

```
MMEMory:LIST?
  ["<path>"]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<path>`	String		Directory to list, absolute and in quotes; see the path model at the beginning of this chapter. Omitted: the root directory of USB:.

Returns

`<String>,<CRD>,<NR1>,...`	One triplet per entry, comma-separated: the entry name in quotes, `FILE` for a regular file or `DIR` for a subdirectory, and the size in bytes. The name is the immediate entry name, not a full path, and a directory always reports the size 0. An empty directory returns a single empty string.

Errors

CODE	MESSAGE	CONDITION
-250	"Mass storage error"	The directory could not be read because of another filesystem failure.
-251	"Missing mass storage"	The path names a storage volume that is not available on the instrument.
-252	"Missing media"	No USB mass-storage medium is present.
-253	"Corrupt media"	The medium is unusable and could not be mounted.
-256	"File name not found"	The directory does not exist.
-257	"File name error"	The path is syntactically invalid, identifies a file instead of a directory, or is not supported by the storage.

Example

```
MMEM:LIST?
-> ”dac”,DIR,0,”log.csv”,FILE,4096
MMEM:LIST? ”/dac”
-> ”test.csv”,FILE,120
MMEM:LIST? ”/dac/empty”
-> ””
```

Related commands

MMEMory:DATA[:DEFinite], MMEMory:DELete, MMEMory:FREE?

11

CHAPTER 11

OUTPut commands

This chapter describes the output control: the permitted power-flow direction of a channel, the inhibit input that acts on all channels, the aggregate clear of the protection and fault latches and the output enable. The voltage setpoint, the current limit and the voltage-sense selection are SOURce commands; the individual protection functions are described there as well.

IN THIS CHAPTER

11.1 OUTPut:INHibit:ACTive 141

11.2 OUTPut:INHibit:MODE 142

11.3 OUTPut:MODE 144

11.4 OUTPut:PROTection:CLEar 145

11.5 OUTPut[:STATe] 146

11.1 OUTPut:INHibit:ACTive

Sets or queries the level at which the inhibit input is active.

The inhibit input is the Digital-I/O line routed with DIGital:LINE<n>:MODE INHibit; the setting has no channel list because the inhibit acts on all channels. With HIGH the inhibit is active while the line reads High, so a released or unconnected line inhibits and a switch to ground releases; LOW is the opposite. What an active inhibit does is selected with OUTPut:INHibit:MODE; in mode OFF the level has no effect. The inhibit is level-based: in LATChing mode a change of the level latches INHIBIT at once when the line already reads the new active level. The setting is accepted while no line is routed to INHibit.

COMMAND

```
OUTPut:INHibit:ACTive
  HIGH|LOW
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
HIGH	CPD		The inhibit is active while the line reads High; a released or unconnected line inhibits.
LOW	CPD		The inhibit is active while the line reads Low, for example when it is switched to ground.

QUERY

```
OUTPut:INHibit:ACTive?
```

Returns

<CRD> HIGH or LOW.

Example

```
OUTP:INH:ACT LOW
OUTP:INH:ACT?
-> LOW
```

Related commands

OUTPut:INHibit:MODE, DIGital:LINE<n>:MODE, STATus:QUEStionable:CONDition?

11.2 OUTPut:INHibit:MODE

Selects or queries how the inhibit input acts on the outputs.

The inhibit acts on all channels: an active inhibit disables every enabled output, and a channel that is already off is unaffected. With LIVE the enabled outputs follow the input, off while it is active and on again when it goes inactive; their commanded state stays ON, and the Questionable condition INHIBIT, bit 4, follows the input. With LATChing an active input switches every output off and latches INHIBIT; the inhibit is level-based, so the latch is also set by an input that is already active when the line is routed, the mode is set to LATChing or OUTPut:INHibit:ACTive is changed, and it stays set while the input stays active, a clear has no effect then; the outputs stay off until OUTPut:PROTection:CLEar has cleared the latch with the input inactive, or *RST has cleared it, and OUTPut ON is required afterwards, as after every protection trip. With OFF the input is ignored while the line stays reserved, and switching to OFF clears no latch. An active or latched inhibit ends DAC playback on every channel abnormally, like OUTPut OFF, and makes INITiate with a TRANsient participant generate -221; an acquisition continues. OUTPut ON is accepted during an active LIVE inhibit and takes effect when the input goes inactive; with a latched inhibit it generates -221. The line is routed with DIGital:LINE<n>:MODE INHibit; without a routed line the mode is accepted and the input counts as inactive, so routing the line to another function re-enables the outputs in LIVE, while a latched INHIBIT stays set until OUTPut:PROTection:CLEar or *RST. The reaction time from the input to the outputs is a product

specification. `OUTPut[:STATe]?` returns the commanded state, 1, while a `LIVE` inhibit holds an enabled output off; the actual state is visible through the `INHIBIT` condition bit.

COMMAND

```
OUTPut:INHibit:MODE
  OFF|LIVE|LATChing
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`OFF`	CPD		The input is ignored; the line stays reserved for the inhibit, and a latched `INHIBIT` is not cleared.
`LIVE`	CPD		The enabled outputs are off while the input is active and on again when it goes inactive.
`LATChing`	CPD		An active input switches every output off and latches `INHIBIT`, also an input that is already active when the mode is set; the latch stays until `OUTPut:PROTection:CLEar` clears it with the input inactive.

QUERY

```
OUTPut:INHibit:MODE?
```

Returns

`<CRD>`	`OFF`, `LIVE` or `LATC`.

Example

```
OUTP:INH:MODE LATC
OUTP:INH:MODE?
-> LATC
```

Related commands

`OUTPut:INHibit:ACTive`, `OUTPut:PROTection:CLEar`, `DIGital:LINE<n>:MODE`, `STATus:QUEStionable:CONDition?`

11.3 OUTPut:MODE

Sets or queries the permitted power-flow direction of a channel.

SOURce permits only power delivery to the DUT, SINK permits only power absorption from the DUT, so the channel works as an active load, and AUTO permits both; the actual direction then results from the operating point. In every mode the voltage loop regulates to the voltage setpoint and the current loop limits at the current limit, so the mode is independent of the CV/CC regulation state. Changing the mode does not change the setpoint, the current limit, the measurement ranges or the output state. On a channel with [SOURce:]ARB[:STATe] ON the mode is locked while an operation with a TRANsient participant is armed or active. The query returns the configured permission, not the momentary direction; the momentary direction is reported by the SOURCING and SINKING bits of the Operation status. The behaviour when the operating point requires a direction that SOURce or SINK forbids is not yet specified and is not to be assumed to be a protection trip or an automatic output shutdown (TBD).

COMMAND

```
OUTPut:MODE
  AUTO|SINK|SOURce[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
AUTO	CPD		Bidirectional operation: source and sink are both permitted.
SINK	CPD		Power absorption only.
SOURce	CPD		Power delivery only.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same mode is applied to every listed channel.

QUERY

```
OUTPut:MODE?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<CRD> AUTO, SINK or SOUR: the configured permission, not the momentary direction of power flow.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has [SOURce:]ARB[:STATe] ON while an operation with a TRANsient participant is armed or active. The mode is unchanged.
-222	"Data out of range"	A listed channel does not exist. The mode is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The mode is unchanged.

Example

```
OUTP:MODE SINK
OUTP:MODE?
-> SINK
OUTP:MODE AUTO,(@2)
OUTP:MODE? (@1:2)
-> SINK,AUTO
```

Related commands

OUTPut[:STATe], [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude], STATus:OPERation:CONDition?

11.4 OUTPut:PROTection:CLEar

Clears the latched OVP, OCP, OPP, OTP, inhibit and hardware-fault trips of a channel in one operation.

The OVP, OCP and OPP latches are cleared as by the three individual ...:PROTection:CLEar commands; besides *RST it is the only clear command for the OTP, INHIBIT and HW_FAULT latches. The protection state, level and delay settings are unchanged, and the output is not re-enabled; it stays off until it is enabled again with OUTPut ON. Latches that are not set are left alone and cause no error, so the command recovers from a trip whose cause has not been identified yet. OTP, INHIBIT and HW_FAULT are cleared only once their cause is gone: OTP once every channel is back in its normal temperature range, INHIBIT once the inhibit input is inactive, which it also is while no line is routed to INHibit, and HW_FAULT once the hardware has recovered by itself, with the internal supply and the channel control path back and the critical self-test passed. While its cause persists such a latch stays set; the clear has no effect on it then and records no new Questionable event. An already latched Questionable status event is not erased, and a trip whose electrical cause is still present prevents enabling the output after the clear. On a channel with [SOURce:]ARB[:STATe] ON the command is locked while an operation with a TRANsient participant is armed or active. The device-wide latches OTP, INHIBIT and HW_FAULT are cleared for every channel whichever channel list is given.

COMMAND

```
OUTPut:PROTection:CLEar
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to clear. Omitted: channel 1. Every listed channel is cleared.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has [SOURce:]ARB[:STATe] ON while an operation with a TRANsient participant is armed or active. No latch is cleared.
-222	"Data out of range"	A listed channel does not exist. No latch is cleared.
-171	"Invalid expression"	A channel appears twice in the channel list. No latch is cleared.

Example

```
VOLT:PROT:TRIP?
-> 1
OUTP:PROT:CLE
VOLT:PROT:TRIP?
-> 0
OUTP ON
```

Related commands

OUTPut[:STATe], STATus:QUEStionable:CONDition?, OUTPut:INHibit:MODE, [SOURce:]VOLTage:PROTection:CLEar

11.5 OUTPut[:STATe]

Enables or disables the output of a channel, or queries its state.

ON enables the channel with the configured voltage setpoint, current limit, output mode, sense selection and protection settings. OFF makes the output stage electronically high-impedance: source and sink are both deactivated. There is no output relay; the terminals are neither isolated nor disconnected, and an externally applied voltage stays present and measurable. Measurements remain available while the output is off and report the actual terminal quantities. OFF is accepted at any time; on a channel whose DAC playback is armed or running it ends the playback of every participating channel abnormally. ON generates -221 while a protection trip, overtemperature, a latched inhibit or a hardware fault is latched, and on a channel with [SOURce:]ARB[:STATe] ON while an operation with a TRANsient participant is armed or active; during an active LIVE inhibit it is accepted and takes effect when the inhibit input goes inactive. The query returns the enable state only; a channel switched off by a latched protection trip reports 0. CV, CC and the direction of power flow are reported in the Operation status.

COMMAND

```
OUTPut[:STATe]
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ON | 1`	Boolean		Enable the output.
`OFF | 0`	Boolean		Disable the output.
`<chanlist>`	Channel list		Channels to switch. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
OUTPut[:STATe]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR1>` 1 when the output is enabled, otherwise `0`.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	A latched protection trip, overtemperature, a latched inhibit or a hardware fault prevents enabling the channel, or `ON` addresses a channel with `[SOURce:]ARB[:STATe] ON` while an operation with a `TRANsient` participant is armed or active. The output stays off.
-222	`"Data out of range"`	A listed channel does not exist. No output is switched.
-171	`"Invalid expression"`	A channel appears twice in the channel list. No output is switched.

Example

```
OUTP ON
OUTP?
-> 1
OUTP ON,(@1:2)
OUTP? (@1:2)
-> 1,1
```

Related commands

OUTPut:MODE, OUTPut:PROTection:CLEar, [SOURce:]VOLTage:SENSe[:SOURce], MEASure[:SCALar]:VOLTage[:DC]?

12

CHAPTER 12

SENSe commands

This chapter describes the measurement path of a channel and the configuration of the acquisition engine. The RANGe subtrees select the voltage and current measurement ranges, manually or by autoranging; they act on the measurement only and never limit a programmed setpoint. The FUNCtion commands select per channel which quantities the acquisition record retains, and the SWEep commands hold the common acquisition timeline: the sample period on the 1 us system timebase, the total number of acquisition points and the position of the window relative to the trigger instant t0. The acquisition is armed with INITiate, read back with FETCh:ARRay and stopped with ABORt; the whole FUNCtion and SWEep configuration is frozen while any operation is armed or active, whether or not it has an ADC participant.

IN THIS CHAPTER

12.1 SENSe:CURRent:RANGe 150

12.2 SENSe:CURRent:RANGe:AUTO 152

12.3 SENSe:FUNCtion:CURRent 153

12.4 SENSe:FUNCtion:VOLTage 155

12.5 SENSe:SWEep:OFFSet:POINts 156

12.6 SENSe:SWEep:POINts 158

12.7 SENSe:SWEep:TINTerval 159

12.8 SENSe:VOLTage:RANGe 161

12.9 SENSe:VOLTage:RANGe:AUTO 163

12.1 SENSe:CURRent:RANGe

Selects a fixed current measurement range of a channel, or queries the active range.

A numeric value is a current in amperes and must be strictly greater than zero; the instrument selects the smallest supported current measurement range whose positive full-scale value is greater than or equal to it, so the request is a signal magnitude, not a range name. Any selection through this command switches current autorange off and keeps the selected range fixed until autorange is enabled again. The measurement range never changes a source range or the programmed current limit, and a limit larger than the selected range stays valid: readings beyond the active range are returned as `9.900000000E+37` or `-9.900000000E+37` and set `READING_OVERFLOW` in the Measurement status. `DEFault` is not accepted, because the default measurement state is autorange rather than a particular fixed range. On a channel with `[SOURce:]ARB[:STATe] ON` the setting is locked from a successful `INITiate` with `TRANsient` until the operation has ended. The query returns the positive full-scale value of the range that is active in hardware at that moment, which with autorange enabled follows the measured signal; it is not a measurement.

COMMAND

```
SENSe:CURRent:RANGe
  <range>|MINimum|MAXimum[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<range>`	NRf, A	Model-dependent	Largest current to be measured. The smallest range that covers the value is selected; autorange is switched off.
`MINimum`	CPD		Select the smallest supported current measurement range.
`MAXimum`	CPD		Select the largest supported current measurement range.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
SENSe:CURRent:RANGe?
  [MINimum|MAXimum][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the full-scale value of the smallest supported range instead of the active one.

`MAXimum`	CPD	Return the full-scale value of the largest supported range instead of the active one.
`<chanlist>`	Channel list	Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, A`	The positive full-scale value of the currently active current measurement range, or of the selected capability range. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has `ARB[:STATe] ON` while a DAC playback is armed or running. The active range and the autorange state are unchanged on every channel.
-222	"Data out of range"	`<range>` is zero, negative or larger than the largest supported range, or a listed channel does not exist. The active range and the autorange state are unchanged.
-224	"Illegal parameter value"	`DEFault` is not accepted by this command. The active range and the autorange state are unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The active range is unchanged.

Example

```
SENS:CURR:RANG 0.05
SENS:CURR:RANG:AUTO?
-> 0
SENS:CURR:RANG MAX,(@2)
SENS:CURR:RANG:AUTO? (@1:2)
-> 0,0
```

Related commands

`SENSe:CURRent:RANGe:AUTO`, `SENSe:VOLTage:RANGe`, `MEASure[:SCALar]:CURRent[:DC]?`, `STATus:MEASurement:CONDition?`

12.2 SENSe:CURRent:RANGe:AUTO

Enables or disables current measurement autoranging of a channel, or queries its state.

With autorange enabled the instrument selects the current measurement range from the magnitude of the measured current and may use hysteresis and timing to avoid range chatter; the programmed current limit is never the ranging criterion. OFF freezes the range that is active at that moment as the fixed range, and does not return to an earlier manually selected range. An explicit SENSe:CURRent:RANGe switches autorange off in the same way. Voltage and current autoranging are independent of each other, and neither affects a source range or a programmed setpoint. On a channel with [SOURce:]ARB[:STATe] ON the setting is locked from a successful INITiate with TRANsient until the operation has ended; an enabled autorange keeps switching ranges meanwhile. The query reports the mode, not the active range, which is read with SENSe:CURRent:RANGe?.

COMMAND

```
SENSe:CURRent:RANGe:AUTO
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		The range follows the measured current.
OFF | 0	Boolean		The currently active range is frozen as the fixed range.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
SENSe:CURRent:RANGe:AUTO?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when current autoranging is enabled, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON while a DAC playback is armed or running. The autorange state is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The autorange state is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The autorange state is unchanged.

Example

```
SENS:CURR:RANG:AUTO ON
SENS:CURR:RANG:AUTO?
-> 1
SENS:CURR:RANG:AUTO OFF,(@2)
SENS:CURR:RANG:AUTO? (@1:2)
-> 1,0
```

Related commands

SENSe:CURRent:RANGe, SENSe:VOLTage:RANGe:AUTO, MEASure[:SCALar]:CURRent[:DC]?

12.3 SENSe:FUNCtion:CURRent

Selects or queries whether the current of a channel is written into the acquisition record.

The selection controls retention only: every channel is always measured at the common acquisition timing, and monitoring, statistics, scalar MEASure queries, regulation and protection are unaffected in either state. Together with SENSe:FUNCtion:VOLTage it defines the width of one acquisition point and therefore the local record capacity reported by FETCh:ARRay:POINts? MAXimum; a point holds the retained quantities in ascending channel order, voltage before current, without placeholders for omitted quantities. A channel with both quantities OFF is not recorded at all, and an INITiate with an ACQuire or DLOG participant while no quantity is ON on any channel generates -221 and arms nothing. The layout is fixed for the lifetime of a record: a completed record keeps the layout it was acquired with, and a later change never reinterprets it. The setting is frozen while any operation is armed or active, whether or not it has an ADC participant, on every channel and whether or not that channel is recorded.

COMMAND

```
SENSe:FUNCtion:CURRent
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ON | 1`	Boolean		The current of the channel is retained in the acquisition record.
`OFF | 0`	Boolean		The current is measured as before but not written into the record.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
SENSe:FUNCtion:CURRent?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR1>` 1 when the current of the channel is recorded, otherwise `0`.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with an ADC participant is armed or active. The selection is unchanged on every channel.
-222	`"Data out of range"`	A listed channel does not exist. The selection is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The selection is unchanged.

Example

```
SENS:FUNC:CURR ON
SENS:FUNC:CURR?
-> 1
SENS:FUNC:CURR OFF,(@2)
SENS:FUNC:CURR? (@1:2)
-> 1,0
```

Related commands

`SENSe:FUNCtion:VOLTage`, `FETCh:ARRay?`, `FETCh:ARRay:POINts?`, `INITiate`

12.4 SENSe:FUNCtion:VOLTage

Selects or queries whether the voltage of a channel is written into the acquisition record.

The selection controls retention only: every channel is always measured at the common acquisition timing, and monitoring, statistics, scalar MEASure queries, regulation and protection are unaffected in either state. A channel may therefore contribute its voltage only, its current only, a coherent voltage and current pair, or nothing; with both quantities OFF the channel is not recorded. The selection changes the width of one acquisition point and therefore the local record capacity, but never the acquisition timing, which is common to all channels. An INITiate with an ACQuire or DLOG participant while no quantity is ON on any channel generates -221 and arms nothing. The setting is frozen while any operation is armed or active, whether or not it has an ADC participant, on every channel and whether or not that channel is recorded.

COMMAND

```
SENSe:FUNCtion:VOLTage
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		The voltage of the channel is retained in the acquisition record.
OFF | 0	Boolean		The voltage is measured as before but not written into the record.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
SENSe:FUNCtion:VOLTage?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when the voltage of the channel is recorded, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active. The selection is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The selection is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The selection is unchanged.

Example

```
SENS:FUNC:VOLT ON
SENS:FUNC:VOLT?
-> 1
SENS:FUNC:VOLT OFF,(@2)
SENS:FUNC:VOLT? (@1:2)
-> 1,0
```

Related commands

SENSe:FUNCtion:CURRent, FETCh:ARRay?, FETCh:ARRay:POINts?, INITiate

12.5 SENSe:SWEep:OFFSet:POINts

Sets or queries the position of the acquisition window relative to the trigger instant t0.

A negative value is a pre-trigger offset: the window begins that many samples before t0, and the instrument keeps that many samples of the running sample stream as an internal history while it waits for the trigger. Zero starts the window at t0, and a positive value is a post-trigger delay that discards the samples between t0 and the window start; DAC playback in the same operation always starts at t0. A pre-trigger offset requires a trigger source other than IMMediate, a depth the history capacity supports and, with a finite target, a depth smaller than SWEep:POINts; the setter does not check this against SWEep:POINts, so the two may be sent in either order, and INITiate reports an inconsistent combination as -221. If the trigger arrives before the requested history has accumulated, it is accepted at once, the available history is kept and the post-trigger part keeps its full length, so the record may be shorter than the target without an error. The setting is frozen while any operation is armed or active, whether or not it has an ADC participant.

COMMAND

```
SENSe:SWEep:OFFSet:POINts
  <points>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<points>	NR1	MINimum to MAXimum (TBD)	Window position in acquisition points: negative for pre-trigger, 0 for a window starting at t0, positive for a post-trigger delay.
MINimum	CPD		Deepest supported pre-trigger offset, limited by the local history capacity and the configured target count.
MAXimum	CPD		Largest supported post-trigger delay in points (TBD).
DEFault	CPD		0: the window starts at t0.

QUERY

```
SENSe:SWEep:OFFSet:POINts?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the deepest supported pre-trigger offset instead of the setting.
MAXimum	CPD		Return the largest supported post-trigger delay instead of the setting.
DEFault	CPD		Return 0 instead of the setting.

Returns

<NR1> The configured window position in acquisition points, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active. The offset is unchanged.
-222	"Data out of range"	<points> is outside the representable offset domain. The offset is unchanged.

Example

```
SENS:SWE:POIN 100000
SENS:SWE:OFFS:POIN -20000
SENS:SWE:OFFS:POIN?
-> -20000
SENS:SWE:OFFS:POIN? DEF
-> 0
```

Related commands

SENSe:SWEep:POINts, SENSe:SWEep:TINTerval, TRIGger[:SEQuence]:SOURce, INITiate

12.6 SENSe:SWEep:POINts

Sets or queries the total number of acquisition time points to produce.

One acquisition point holds every retained quantity of every recorded channel and is never counted as several points, so the target is independent of the SENSe:FUNCtion selection. The target may exceed the number of points that fit simultaneously in the internal record; such an acquisition completes only when FETCh:ARRay:NEXT? consumes blocks fast enough to free space, and otherwise ends with an overrun before unread points would be overwritten. INFinity selects an unbounded acquisition that never ends by reaching a count and stops only through ABORt or a terminal fault; it is the intended setting for a DLog run that logs until stopped, and with INITiate ACQuire it requires a consuming reader. With an unbounded target an *OPC? or *WAI after INITiate waits until ABORt, a terminal fault or *RST ends the acquisition. MAXimum is not accepted, because the counter limit says nothing about record capacity, which is queried with FETCh:ARRay:POINts? MAXimum. The setting is frozen while any operation is armed or active, whether or not it has an ADC participant, and the query returns the configured target, not the number of points already recorded.

COMMAND

```
SENSe:SWEep:POINts
  <count>|MINimum|DEFault|INFinity
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<count>	NR1	1 to TBD	Total number of acquisition time points. The product guarantees at least 4 000 000 coherent voltage and current points in the internal record; a larger target needs a consuming reader.
MINimum	CPD		1, the shortest acquisition.
DEFault	CPD		1.
INFinity	CPD		Unbounded acquisition. Character data, not a numeric value.

QUERY

```
SENSe:SWEep:POINts?
  [MINimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return 1 instead of the setting.
DEFault	CPD		Return 1 instead of the setting.

Returns

<NR1> or <CRD> The configured target count, or INF when the unbounded target is selected.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active. The target count is unchanged.
-222	"Data out of range"	<count> cannot be represented by the acquisition counter. The target count is unchanged.
-224	"Illegal parameter value"	MAXimum is not accepted by this command. The target count is unchanged.

Example

```
SENS:SWE:POIN 100000
SENS:SWE:POIN?
-> 100000
SENS:SWE:POIN INF
SENS:SWE:POIN?
-> INF
```

Related commands

SENSe:SWEep:TINTerval, SENSe:SWEep:OFFSet:POINts, FETCh:ARRay:POINts?, INITiate

12.7 SENSe:SWEep:TINTerval

Sets or queries the interval between two consecutive acquisition time points.

The interval is the common acquisition timeline of the whole instrument: it applies to every channel, whether or not its quantities are retained, and there is no per-channel timing and no second RATE command for the same setting. It lies on the 1 us system timebase that the acquisition shares with the DAC, whose update period [SOURce:]ARB:TINTerval uses the coarser 10 us grid: a requested value is quantized to the nearest multiple of 1 us, with exact halfway cases rounded to the larger one, and the query returns the quantized value. Together with the target count it defines the duration of the acquisition, and it is the only source of the relative timing of the record, which carries no per-point timestamps. The setting is frozen while any operation is armed or active, whether or not it has an ADC participant.

COMMAND

```
SENSe:SWEep:TINTerval
  <seconds>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	2 us to MAXimum (TBD)	New acquisition interval. Quantized to the nearest multiple of 1 us.
`MINimum`	CPD		2 us, the maximum acquisition rate of 500 kSa/s: every channel measures voltage and current at every point.
`MAXimum`	CPD		Longest programmable interval (TBD).
`DEFault`	CPD		10 us.

QUERY

```
SENSe:SWEep:TINTerval?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the shortest interval, 2 us, instead of the setting.
`MAXimum`	CPD		Return the longest programmable interval instead of the setting.
`DEFault`	CPD		Return 10 us instead of the setting.

Returns

`<NR3>, s`	The configured, quantized acquisition interval, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active. The interval is unchanged.
-222	"Data out of range"	`<seconds>` is outside `MINimum` to `MAXimum`. The interval is unchanged.
-222	"Data out of range"	`<interval>` is below `MINimum` or above `MAXimum`; a value below the minimum is not rounded up. The interval is unchanged.

Example

```
SENS:SWE:TINT 100E-6
SENS:SWE:TINT?
-> 1.000000000E-04
SENS:SWE:TINT 2.5 us
SENS:SWE:TINT?
-> 3.000000000E-06
SENS:SWE:TINT? MIN
-> 2.000000000E-06
```

Related commands

`SENSe:SWEep:POINts`, `SENSe:SWEep:OFFSet:POINts`, `[SOURce:]ARB:TINTerval`, `INITiate`

12.8 SENSe:VOLTage:RANGe

Selects a fixed voltage measurement range of a channel, or queries the active range.

A numeric value is a voltage in volts and must be strictly greater than zero; the instrument selects the smallest supported voltage measurement range whose positive full-scale value is greater than or equal to it, so the request is a signal magnitude, not a range name. Any selection through this command switches voltage autorange off and keeps the selected range fixed until autorange is enabled again. The measurement range never changes a source range or the programmed voltage setpoint, and a setpoint larger than the selected range stays valid: readings beyond the active range are returned as `9.900000000E+37` or `-9.900000000E+37` and set `READING_OVERFLOW` in the Measurement status. `DEFault` is not accepted, because the default measurement state is autorange rather than a particular fixed range. On a channel with `[SOURce:]ARB[:STATe] ON` the setting is locked from a successful `INITiate` with `TRANsient` until the operation has ended. The query returns the positive full-scale value of the range that is active in hardware at that moment, which with autorange enabled follows the measured signal; it is not a measurement.

COMMAND

```
SENSe:VOLTage:RANGe
  <range>|MINimum|MAXimum[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<range>`	NRf, V	Model-dependent	Largest voltage to be measured. The smallest range that covers the value is selected; autorange is switched off.
`MINimum`	CPD		Select the smallest supported voltage measurement range.

`MAXimum`	CPD	Select the largest supported voltage measurement range.
`<chanlist>`	Channel list	Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
SENSe:VOLTage:RANGe?
  [MINimum|MAXimum][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the full-scale value of the smallest supported range instead of the active one.
`MAXimum`	CPD		Return the full-scale value of the largest supported range instead of the active one.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` — The positive full-scale value of the currently active voltage measurement range, or of the selected capability range. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	A listed channel has `ARB[:STATe] ON` while a DAC playback is armed or running. The active range and the autorange state are unchanged on every channel.
-222	`"Data out of range"`	`<range>` is zero, negative or larger than the largest supported range, or a listed channel does not exist. The active range and the autorange state are unchanged.
-224	`"Illegal parameter value"`	`DEFault` is not accepted by this command. The active range and the autorange state are unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The active range is unchanged.

Example

```
SENS:VOLT:RANG 5
SENS:VOLT:RANG:AUTO?
-> 0
SENS:VOLT:RANG MIN,(@2)
SENS:VOLT:RANG:AUTO? (@1:2)
-> 0,0
```

Related commands

SENSe:VOLTage:RANGe:AUTO, SENSe:CURRent:RANGe, MEASure[:SCALar]:VOLTage[:DC]?, STATus:MEASurement:CONDition?

12.9 SENSe:VOLTage:RANGe:AUTO

Enables or disables voltage measurement autoranging of a channel, or queries its state.

With autorange enabled the instrument selects the voltage measurement range from the measured voltage and may use hysteresis and timing to avoid range chatter; the programmed voltage setpoint is never the ranging criterion. OFF freezes the range that is active at that moment as the fixed range, and does not return to an earlier manually selected range. An explicit SENSe:VOLTage:RANGe switches autorange off in the same way. Voltage and current autoranging are independent of each other, and neither affects a source range or a programmed setpoint. On a channel with [SOURce:]ARB[:STATe] ON the setting is locked from a successful INITiate with TRANsient until the operation has ended; an enabled autorange keeps switching ranges meanwhile. The query reports the mode, not the active range, which is read with SENSe:VOLTage:RANGe?.

COMMAND

```
SENSe:VOLTage:RANGe:AUTO
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		The range follows the measured voltage.
OFF | 0	Boolean		The currently active range is frozen as the fixed range.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
SENSe:VOLTage:RANGe:AUTO?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when voltage autoranging is enabled, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON while a DAC playback is armed or running. The autorange state is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The autorange state is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The autorange state is unchanged.

Example

```
SENS:VOLT:RANG:AUTO ON
SENS:VOLT:RANG:AUTO?
-> 1
SENS:VOLT:RANG:AUTO OFF,(@2)
SENS:VOLT:RANG:AUTO? (@1:2)
-> 1,0
```

Related commands

SENSe:VOLTage:RANGe, SENSe:CURRent:RANGe:AUTO, MEASure[:SCALar]:VOLTage[:DC]?

13

CHAPTER 13

SOURce commands

This chapter describes the source configuration of a channel: the voltage setpoint, the current limit and the voltage-sense selection. The instrument regulates the voltage and limits the current; CV and CC are resulting operating states reported in the Operation status, not selectable modes. The protection functions OVP, OCP and OPP each have their own PROTection subtree with enable state, threshold, trip query and clear; a protection trip switches the output off and is distinct from reaching the current limit. The output enable and the power-flow mode are OUTPut commands. The ARB subtree configures arbitrary waveform playback of the DAC: a global mode and update period, per-channel participation, shape, sequence and termination settings, and the host-fed stream; playback is armed with INITiate TRANsient and stopped with ABORt.

IN THIS CHAPTER

13.1 [SOURce:]ARB:EXPonential:END:TIME 168
13.2 [SOURce:]ARB:EXPonential:END[:LEVel] 169
13.3 [SOURce:]ARB:EXPonential:STARt:TIME 171
13.4 [SOURce:]ARB:EXPonential:STARt[:LEVel] 173
13.5 [SOURce:]ARB:EXPonential:TCONstant 174
13.6 [SOURce:]ARB:EXPonential:TIME 176
13.7 [SOURce:]ARB:FUNCtion:SHAPe 178
13.8 [SOURce:]ARB:LOGStaircase:END:TIME 179
13.9 [SOURce:]ARB:LOGStaircase:END[:LEVel] 181
13.10 [SOURce:]ARB:LOGStaircase:NSTeps 183
13.11 [SOURce:]ARB:LOGStaircase:STARt:TIME 184
13.12 [SOURce:]ARB:LOGStaircase:STARt[:LEVel] 186
13.13 [SOURce:]ARB:LOGStaircase:TIME 188
13.14 [SOURce:]ARB:MODE 189
13.15 [SOURce:]ARB:PULSe:HIGH 190
13.16 [SOURce:]ARB:PULSe:HIGH:TIME 192
13.17 [SOURce:]ARB:PULSe:LOW 194

13.18 [SOURce:]ARB:PULSe:LOW:TIME 195

13.19 [SOURce:]ARB:RAMP:END:TIME 197

13.20 [SOURce:]ARB:RAMP:END[:LEVel] 199

13.21 [SOURce:]ARB:RAMP:RTIMe 200

13.22 [SOURce:]ARB:RAMP:STARt:TIME 202

13.23 [SOURce:]ARB:RAMP:STARt[:LEVel] 204

13.24 [SOURce:]ARB:SEQuence:COUNt 205

13.25 [SOURce:]ARB:SEQuence:DATA 207

13.26 [SOURce:]ARB:SEQuence:DATA:POINts? 209

13.27 [SOURce:]ARB:SEQuence:FILE 210

13.28 [SOURce:]ARB:SEQuence:PAUSe 212

13.29 [SOURce:]ARB:SEQuence:SOURce 213

13.30 [SOURce:]ARB:SINusoid:AMPLitude 215

13.31 [SOURce:]ARB:SINusoid:FREQuency 217

13.32 [SOURce:]ARB:SINusoid:OFFSet 218

13.33 [SOURce:]ARB:SINusoid:PHASe 220

13.34 [SOURce:]ARB:STAircase:END:TIME 222

13.35 [SOURce:]ARB:STAircase:END[:LEVel] 223

13.36 [SOURce:]ARB:STAircase:NSTeps 225

13.37 [SOURce:]ARB:STAircase:STARt:TIME 227

13.38 [SOURce:]ARB:STAircase:STARt[:LEVel] 228

13.39 [SOURce:]ARB:STAircase:TIME 230

13.40 [SOURce:]ARB:STReam:DATA 232

13.41 [SOURce:]ARB:STReam:DATA:CURRent 233

13.42 [SOURce:]ARB:STReam:DATA:VOLTage 234

13.43 [SOURce:]ARB:STReam:POINts 236

13.44 [SOURce:]ARB:TERMinate:LAST 237

13.45 [SOURce:]ARB:TINTerval 238

13.46 [SOURce:]ARB:TRAPezoid:END:TIME 240

13.47 [SOURce:]ARB:TRAPezoid:FTIMe 241

13.48 [SOURce:]ARB:TRAPezoid:RTIMe 243

13.49 [SOURce:]ARB:TRAPezoid:STARt:TIME 245

13.50 [SOURce:]ARB:TRAPezoid:STARt[:LEVel] 246

13.51 [SOURce:]ARB:TRAPezoid:TOP:TIME 248

13.52 [SOURce:]ARB:TRAPezoid:TOP[:LEVel] 249

13.53 [SOURce:]ARB:TRIangle:FTIMe 251

13.54 [SOURce:]ARB:TRIangle:RTIMe 253

13.55 [SOURce:]ARB:TRIangle:STARt[:LEVel] 254

13.56 [SOURce:]ARB:TRIangle:TOP[:LEVel] 256

13.57 [SOURce:]ARB[:STATe] 258

13.58 [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] 259

13.59 [SOURce:]CURRent:PROTection:CLEar 261

13.60 [SOURce:]CURRent:PROTection:DELay 262

13.61 [SOURce:]CURRent:PROTection:LEVel 264

13.62 [SOURce:]CURRent:PROTection:TRIPped? 265

13.63 [SOURce:]CURRent:PROTection[:STATe] 266

13.64 [SOURce:]POWer:PROTection:CLEar 268

13.65 [SOURce:]POWer:PROTection:DELay 269

13.66 [SOURce:]POWer:PROTection:LEVel 270

13.67 [SOURce:]POWer:PROTection:TRIPped? 272

13.68 [SOURce:]POWer:PROTection[:STATe] 273

13.69 [SOURce:]VOLTage:PROTection:CLEar 274

13.70 [SOURce:]VOLTage:PROTection:LEVel 275

13.71 [SOURce:]VOLTage:PROTection:TRIPped? 277

13.72 [SOURce:]VOLTage:PROTection[:STATe] 278

13.73 [SOURce:]VOLTage:SENSe[:SOURce] 280

13.74 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude] 281

13.1 [SOURce:]ARB:EXPonential:END:TIME

Sets or queries how long the exponential shape holds its end level after the transition.

One execution of the shape is: hold ARB:EXPonential:STARt for ARB:EXPonential:STARt:TIME, run the exponential transition for ARB:EXPonential:TIME, then hold ARB:EXPonential:END for this time, so the total execution lasts STARt:TIME + TIME + END:TIME. The hold begins only after the transition time has elapsed and the output has been set exactly to the end level. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept, so a later change of the update period changes only how many DAC points the hold produces. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe EXPonential in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:EXPonential:END:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time of the end level at the end of one execution. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:END:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed hold time of the end level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The hold time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:EXP:END:TIME 0.2
ARB:EXP:END:TIME?
-> 2.000000000E-01
ARB:EXP:END:TIME 0.5,(@2)
ARB:EXP:END:TIME? (@1:2)
-> 2.000000000E-01,5.000000000E-01
```

Related commands

[SOURce:]ARB:EXPonential:END[:LEVel], [SOURce:]ARB:EXPonential:TIME, [SOURce:]ARB:EXPonential:STARt:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.2 [SOURce:]ARB:EXPonential:END[:LEVel]

Sets or queries the level the exponential shape reaches at the end of its transition.

In one execution the exponential transition runs from ARB:EXPonential:STARt towards this level for exactly ARB:EXPonential:TIME; at the end of that time the output is set exactly to the end level and then held for ARB:EXPonential:END:TIME. Completion therefore never depends on numerical convergence to the mathematical asymptote of the curve. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its [SOURce:]CURRent value and takes the regulator into CC if the load reaches it. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe EXPonential in ARB:MODE SEQuence. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:EXPonential:END[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint reached at the end of the transition and held afterwards.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:END[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed end level of the exponential shape, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The end level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the capability of the channel, or a listed channel does not exist. The end level is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The end level is unchanged.

Example

```
ARB:EXP:END 5
ARB:EXP:END?
-> 5.000000000E+00
ARB:EXP:END 2,(@2)
ARB:EXP:END? (@1:2)
-> 5.000000000E+00,2.000000000E+00
```

Related commands

`[SOURce:]ARB:EXPonential:STARt[:LEVel]`, `[SOURce:]ARB:EXPonential:TIME`, `[SOURce:]ARB:EXPonential:END:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.3 [SOURce:]ARB:EXPonential:STARt:TIME

Sets or queries how long the exponential shape holds its start level before the transition.

One execution begins with this hold of `ARB:EXPonential:STARt`; the exponential transition to `ARB:EXPonential:END` follows and lasts `ARB:EXPonential:TIME`, then the end level is held for `ARB:EXPonential:END:TIME`. The hold is part of the waveform and is repeated with every execution, unlike the pause of `ARB:SEQuence:PAUSe`, which lies between executions. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe EXPonential` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:EXPonential:STARt:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Hold time of the start level at the beginning of one execution. 0 s omits the hold.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 0 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:STARt:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed hold time of the start level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The hold time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:EXP:STAR:TIME 0.1
ARB:EXP:STAR:TIME?
-> 1.000000000E-01
ARB:EXP:STAR:TIME 50 ms,(@2)
ARB:EXP:STAR:TIME? (@1:2)
-> 1.000000000E-01,5.000000000E-02
```

Related commands

[SOURce:]ARB:EXPonential:STARt[:LEVel], [SOURce:]ARB:EXPonential:TIME, [SOURce:]ARB:EXPonential:END:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.4 [SOURce:]ARB:EXPonential:STARt[:LEVel]

Sets or queries the level the exponential shape starts from.

One execution holds this level for `STARt:TIME`, then leaves it along the exponential transition towards the end level, whose curvature `TCONstant` sets. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its `[SOURce:]CURRent` value. Programming the level neither starts playback nor changes the present output, which follows `[SOURce:]VOLTage` until playback begins. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe EXPonential` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:EXPonential:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint held at the beginning of one execution and left along the transition.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V`	The programmed start level of the exponential shape, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The start level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the capability of the channel, or a listed channel does not exist. The start level is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The start level is unchanged.

Example

```
ARB:EXP:STAR 1
ARB:EXP:STAR?
-> 1.000000000E+00
ARB:EXP:STAR 0,(@2)
ARB:EXP:STAR? (@1:2)
-> 1.000000000E+00,0.000000000E+00
```

Related commands

`[SOURce:]ARB:EXPonential:END[:LEVel]`, `[SOURce:]ARB:EXPonential:STARt:TIME`, `[SOURce:]ARB:EXPonential:TCONstant`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.5 [SOURce:]ARB:EXPonential:TCONstant

Sets or queries the time constant that shapes the exponential transition.

The time constant defines the curvature of the transition between `ARB:EXPonential:STARt` and `ARB:EXPonential:END`: a small value bends the curve sharply towards the end level early, a time constant much longer than `ARB:EXPonential:TIME` makes the transition nearly linear. It does not define how long the transition lasts; that is `ARB:EXPonential:TIME` alone, at whose end the output is set exactly to the end level. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe EXPonential` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:EXPonential:TCONstant
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Time constant of the exponential curve.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:TCONstant?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed time constant, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time constant is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time constant is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time constant is unchanged.

Example

```
ARB:EXP:TCON 0.05
ARB:EXP:TCON?
-> 5.000000000E-02
ARB:EXP:TCON 10 ms,(@2)
ARB:EXP:TCON? (@1:2)
-> 5.000000000E-02,1.000000000E-02
```

Related commands

[SOURce:]ARB:EXPonential:TIME, [SOURce:]ARB:EXPonential:STARt[:LEVel], [SOURce:]ARB:EXPonential:END[:LEVel], [SOURce:]ARB:FUNCtion:SHAPe

13.6 [SOURce:]ARB:EXPonential:TIME

Sets or queries the duration of the exponential transition from the start level to the end level.

The transition sits between the start hold of ARB:EXPonential:STARt:TIME and the end hold of ARB:EXPonential:END:TIME and lasts exactly this time, so one execution takes STARt:TIME + TIME + END:TIME. The transition is deterministic: at the end of the time the output is set exactly to ARB:EXPonential:END, independent of how far the curve shaped by TCONstant has approached its asymptote. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept, so a shorter update period yields more DAC points for the same transition. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe EXPonential in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:EXPonential:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Duration of the transition from the start level to the end level. 0 s omits the phase.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:EXPonential:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed transition time, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The transition time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The transition time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The transition time is unchanged.

Example

```
ARB:EXP:TIME 1
ARB:EXP:TIME?
-> 1.000000000E+00
ARB:EXP:TIME 0.25,(@2)
ARB:EXP:TIME? (@1:2)
-> 1.000000000E+00,2.500000000E-01
```

Related commands

[SOURce:]ARB:EXPonential:TCONstant, [SOURce:]ARB:EXPonential:END[:LEVel], [SOURce:]ARB:TINTerval, [SOURce:]ARB:SEQuence:COUNt

13.7 [SOURce:]ARB:FUNCtion:SHAPe

Selects or queries the waveform a channel plays in programmed playback.

The selection is a per-channel setting and takes effect in ARB:MODE SEQuence only. The eight native generators modulate the voltage setpoint; the current limit stays at its [SOURce:]CURRent value, and the regulator enters CC as for a static setpoint. SEQuence selects the user-defined segment sequence, which carries a voltage setpoint and a current limit per segment and whose data comes from the CSV file or the in-memory list chosen with ARB:SEQuence:SOURce. Selecting a shape neither starts playback nor enables the output.

COMMAND

```
[SOURce:]ARB:FUNCtion:SHAPe
  PULSe|TRIangle|RAMP|TRAPezoid|SINusoid|STAircase|LOGStaircase|EXPonential|
  SEQuence[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
PULSe	CPD		Low level for a low time, then high level for a high time; ARB:PULSe parameters.
TRIangle	CPD		Rise from a start level to a top level and fall back; ARB:TRIangle parameters.
RAMP	CPD		Hold a start level, ramp to an end level, hold it; ARB:RAMP parameters.
TRAPezoid	CPD		Start hold, rise, top hold, fall, end hold; ARB:TRAPezoid parameters.
SINusoid	CPD		One period of a sine around an offset; ARB:SINusoid parameters.
STAircase	CPD		Linear staircase from a start level to an end level; ARB:STAircase parameters.
LOGStaircase	CPD		Logarithmic staircase from a start level to an end level; ARB:LOGStaircase parameters.
EXPonential	CPD		Exponential transition from a start level to an end level; ARB:EXPonential parameters.
SEQuence	CPD		User-defined segment sequence from the source selected with ARB:SEQuence:SOURce.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same shape is applied to every listed channel.

QUERY

```
[SOURce:]ARB:FUNCtion:SHAPe?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<CRD> PULS, TRI, RAMP, TRAP, SIN, STA, LOGS, EXP or SEQ.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The shape is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The shape is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The shape is unchanged.

Example

```
ARB:FUNC:SHAP SIN
ARB:FUNC:SHAP?
-> SIN
ARB:FUNC:SHAP SEQ,(@2)
ARB:FUNC:SHAP? (@1:2)
-> SIN,SEQ
```

Related commands

[SOURce:]ARB:MODE, [SOURce:]ARB:SEQuence:SOURce, [SOURce:]ARB[:STATe], [SOURce:]ARB:SEQuence:COUNt

13.8 [SOURce:]ARB:LOGStaircase:END:TIME

Sets or queries how long the logarithmic staircase holds its end level after the last step.

One execution of the shape is: hold the start level for STARt:TIME, run NSTeps logarithmically spaced levels over TIME, then hold the end level for this time. The last step is already exactly the end level and is counted inside the staircase time, so this hold extends the end level rather than producing it. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe LOGStaircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:LOGStaircase:END:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Hold time of the end level after the staircase has completed. 0 s omits the hold.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 0 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:END:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 0 s, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s`	The programmed hold time of the end level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The hold time is unchanged on every channel.
-222	`"Data out of range"`	`<seconds>` is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:LOGS:END:TIME 0.2
ARB:LOGS:END:TIME?
-> 2.000000000E-01
ARB:LOGS:END:TIME 1,(@2)
ARB:LOGS:END:TIME? (@1:2)
-> 2.000000000E-01,1.000000000E+00
```

Related commands

`[SOURce:]ARB:LOGStaircase:END[:LEVel]`, `[SOURce:]ARB:LOGStaircase:TIME`, `[SOURce:]ARB:LOGStaircase:STARt:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.9 [SOURce:]ARB:LOGStaircase:END[:LEVel]

Sets or queries the level the last step of the logarithmic staircase reaches.

For step `k = 1...NSTeps` the level is `STARt * (END / STARt)^(k / NSTeps)`, so the last step is exactly this end level and lies inside `TIME`; `END:TIME` holds it afterwards. The progression is only defined when the end level and the start level are both non-zero and have the same sign. The setter accepts every value inside the programmable range and `INITiate` checks the pair on each participating channel, so the two levels can be programmed in either order and an unplayable pair is rejected there instead of being silently played as a linear staircase. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its `[SOURce:]CURRent` value. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe LOGStaircase` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:LOGStaircase:END[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint of the last staircase step. A participating channel needs a non-zero level of the same sign as the start level.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:END[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The programmed end level of the logarithmic staircase, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The end level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the capability of the channel, or a listed channel does not exist; the end level is unchanged. At INITiate, a participating LOGStaircase channel whose start and end level are zero or of opposite sign; nothing is armed.
-171	"Invalid expression"	A channel appears twice in the channel list. The end level is unchanged.

Example

```
ARB:LOGS:END 10
ARB:LOGS:END?
-> 1.000000000E+01
ARB:LOGS:END 2,(@2)
ARB:LOGS:END? (@1:2)
-> 1.000000000E+01,2.000000000E+00
```

Related commands

[SOURce:]ARB:LOGStaircase:STARt[:LEVel], [SOURce:]ARB:LOGStaircase:NSTeps, [SOURce:]ARB:LOGStaircase:END:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.10 [SOURce:]ARB:LOGStaircase:NSTeps

Sets or queries how many logarithmically spaced steps the staircase executes.

The steps are distributed over ARB:LOGStaircase:TIME, each one lasting TIME / NSTeps, and step k has the level STARt * (END / STARt)^(k / NSTeps), so the count sets both the level spacing and the dwell of a single step. The start level is not itself one of the steps: the first step lies between the start and the end level, and the last step is exactly ARB:LOGStaircase:END. At least one step is required, and increasing the count shortens each step without changing the total staircase time. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe LOGStaircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:LOGStaircase:NSTeps
  <count>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<count>	NR1	1 to TBD	Number of logarithmically spaced staircase steps in one execution.
MINimum	CPD		Smallest programmable step count (TBD).
MAXimum	CPD		Largest programmable step count (TBD).
DEFault	CPD		Factory default, 10.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:NSTeps?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable step count instead of the setting.
MAXimum	CPD		Return the largest programmable step count instead of the setting.
DEFault	CPD		Return the factory default, 10, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1>	The programmed number of staircase steps, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The step count is unchanged on every channel.
-222	"Data out of range"	<count> is smaller than 1 or above the programmable maximum, or a listed channel does not exist. The step count is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The step count is unchanged.

Example

```
ARB:LOGS:NST 10
ARB:LOGS:NST?
-> 10
ARB:LOGS:NST 20,(@2)
ARB:LOGS:NST? (@1:2)
-> 10,20
```

Related commands

[SOURce:]ARB:LOGStaircase:TIME, [SOURce:]ARB:LOGStaircase:STARt[:LEVel], [SOURce:]ARB:LOGStaircase:END[:LEVel], [SOURce:]ARB:FUNCtion:SHAPe

13.11 [SOURce:]ARB:LOGStaircase:STARt:TIME

Sets or queries how long the logarithmic staircase holds its start level before the first step.

One execution begins with this hold at the start level; the NSTeps steps over TIME follow, and END:TIME closes the execution. The hold is part of the waveform and occurs in every execution, unlike ARB:SEQuence:PAUSe, which lies between executions. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe LOGStaircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:LOGStaircase:STARt:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time of the start level before the first step. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:STARt:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed hold time of the start level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The hold time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:LOGS:STAR:TIME 0.1
ARB:LOGS:STAR:TIME?
-> 1.000000000E-01
ARB:LOGS:STAR:TIME 20 ms,(@2)
ARB:LOGS:STAR:TIME? (@1:2)
-> 1.000000000E-01,2.000000000E-02
```

Related commands

`[SOURce:]ARB:LOGStaircase:STARt[:LEVel]`, `[SOURce:]ARB:LOGStaircase:TIME`, `[SOURce:]ARB:LOGStaircase:END:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.12 [SOURce:]ARB:LOGStaircase:STARt[:LEVel]

Sets or queries the level the logarithmic staircase starts from.

The level is held for `ARB:LOGStaircase:STARt:TIME` at the beginning of one execution and is the base of the progression `STARt * (END / STARt)^(k / NSTeps)`, so it is not itself one of the `ARB:LOGStaircase:NSTeps` steps. It and `ARB:LOGStaircase:END` must both be non-zero and of the same sign, because a real logarithmic progression is otherwise undefined. The setter accepts every value inside the programmable range and `INITiate` checks the pair on each participating channel, so the two levels can be programmed in either order and an unplayable pair is rejected there instead of falling back to linear spacing. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its `[SOURce:]CURRent` value. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe LOGStaircase` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:LOGStaircase:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint at the beginning of one execution. A participating channel needs a non-zero level of the same sign as the end level.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The programmed start level of the logarithmic staircase, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The start level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the capability of the channel, or a listed channel does not exist; the start level is unchanged. At INITiate, a participating LOGStaircase channel whose start and end level are zero or of opposite sign; nothing is armed.
-171	"Invalid expression"	A channel appears twice in the channel list. The start level is unchanged.

Example

```
ARB:LOGS:STAR 0.1
ARB:LOGS:STAR?
-> 1.000000000E-01
ARB:LOGS:STAR 1,(@2)
ARB:LOGS:STAR? (@1:2)
-> 1.000000000E-01,1.000000000E+00
```

Related commands

[SOURce:]ARB:LOGStaircase:END[:LEVel], [SOURce:]ARB:LOGStaircase:NSTeps, [SOURce:]ARB:LOGStaircase:STARt:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.13 [SOURce:]ARB:LOGStaircase:TIME

Sets or queries the total time the logarithmic staircase needs for all its steps.

The time covers the complete staircase from the first step to the last and is divided equally, so one step lasts `TIME / NSTeps`; it excludes the start hold of `ARB:LOGStaircase:STARt:TIME` and the end hold of `ARB:LOGStaircase:END:TIME`, and one execution therefore takes the sum of the three. Because the last step is already `ARB:LOGStaircase:END`, the end level is reached at the end of this time, not after it. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe LOGStaircase` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:LOGStaircase:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Total time of all staircase steps. 0 s omits the phase.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 1 ms.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:LOGStaircase:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 1 ms, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s`	The programmed total staircase time, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The staircase time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The staircase time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The staircase time is unchanged.

Example

```
ARB:LOGS:TIME 2
ARB:LOGS:TIME?
-> 2.000000000E+00
ARB:LOGS:TIME 0.5,(@2)
ARB:LOGS:TIME? (@1:2)
-> 2.000000000E+00,5.000000000E-01
```

Related commands

[SOURce:]ARB:LOGStaircase:NSTeps, [SOURce:]ARB:LOGStaircase:STARt:TIME, [SOURce:]ARB:LOGStaircase:END:TIME, [SOURce:]ARB:TINTerval

13.14 [SOURce:]ARB:MODE

Selects or queries whether the DAC plays a host-fed stream or a waveform generated by the instrument.

The mode is global: one transient operation runs either the host-fed stream or programmed playback on every participating channel, never both. SEQuence plays the shape selected per channel with ARB:FUNCtion:SHAPe, with repeat count, pause and interpolation; STReam plays the points the controller supplies with ARB:STReam:DATA, without loops, pause or interpolation. Both modes use the common update period ARB:TINTerval. Switching between the two clears the DAC sample buffer and the last-valid stream values; a STReam:DATA command in SEQuence mode is rejected. Selecting the mode does not start playback.

COMMAND

```
[SOURce:]ARB:MODE
  STReam|SEQuence
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
STReam	CPD		Host-fed DAC stream from ARB:STReam:DATA blocks.
SEQuence	CPD		Programmed playback: a native generator shape or a user sequence per channel.

QUERY

```
[SOURce:]ARB:MODE?
```

Returns

<CRD> STR for the host-fed stream, SEQ for programmed playback.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The mode and the DAC sample buffer are unchanged.

Example

```
ARB:MODE STR
ARB:MODE?
-> STR
```

Related commands

[SOURce:]ARB[:STATe], [SOURce:]ARB:STReam:DATA, [SOURce:]ARB:FUNCtion:SHAPe, [SOURce:]ARB:TINTerval

13.15 [SOURce:]ARB:PULSe:HIGH

Sets or queries the high level of the pulse shape of a channel.

The level is the voltage setpoint held during the high phase. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback, and the regulator enters CC as for a static setpoint. One execution holds the low level for ARB:PULSe:LOW:TIME, then the high level for ARB:PULSe:HIGH:TIME. The setting is per channel, takes effect in ARB:MODE SEQuence with ARB:FUNCtion:SHAPe PULSe, and is kept while another shape is selected. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:PULSe:HIGH
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	New voltage setpoint of the high phase.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:PULSe:HIGH?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` — The programmed high level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the programmable range, or a listed channel does not exist. The level is unchanged.

-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:PULS:HIGH 5
ARB:PULS:HIGH?
-> 5.000000000E+00
ARB:PULS:HIGH 3.3,(@2)
ARB:PULS:HIGH? (@1:2)
-> 5.000000000E+00,3.300000000E+00
```

Related commands

[SOURce:]ARB:PULSe:HIGH:TIME, [SOURce:]ARB:PULSe:LOW, [SOURce:]ARB:FUNCtion:SHAPe, [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

13.16 [SOURce:]ARB:PULSe:HIGH:TIME

Sets or queries how long the pulse shape of a channel holds its high level.

One execution holds the low level for ARB:PULSe:LOW:TIME and then the high level for this time, and it ends with the high phase; the length of one execution is the sum of the two times. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. A later change of ARB:TINTerval therefore changes how many DAC points the phase produces, not the pulse. Periodic operation is expressed with ARB:SEQuence:COUNt and ARB:SEQuence:PAUSe; the native interface deliberately has no frequency or duty-cycle command. The setting is per channel.

COMMAND

```
[SOURce:]ARB:PULSe:HIGH:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Duration of the high phase. 0 s omits the phase.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:PULSe:HIGH:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed high time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:PULS:HIGH:TIME 1E-3
ARB:PULS:HIGH:TIME?
-> 1.000000000E-03
ARB:PULS:HIGH:TIME 500 us,(@2)
ARB:PULS:HIGH:TIME? (@1:2)
-> 1.000000000E-03,5.000000000E-04
```

Related commands

[SOURce:]ARB:PULSe:LOW:TIME, [SOURce:]ARB:PULSe:HIGH, [SOURce:]ARB:TINTerval, [SOURce:]ARB:SEQuence:COUNt

13.17 [SOURce:]ARB:PULSe:LOW

Sets or queries the low level of the pulse shape of a channel.

The level is the voltage setpoint one execution starts at and holds for ARB:PULSe:LOW:TIME before the high phase begins. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback. The low level is not required to be below the high level: the two levels are programmed independently, and a low level above the high level simply inverts the pulse. The setting is per channel and takes effect in ARB:MODE SEQuence with ARB:FUNCtion:SHAPe PULSe. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:PULSe:LOW
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<value>	NRf, V	Model-dependent	New voltage setpoint of the low phase.
MINimum	CPD		Smallest programmable level (model-dependent).
MAXimum	CPD		Largest programmable level (model-dependent).
DEFault	CPD		Factory default, 0 V.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:PULSe:LOW?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V`	The programmed low level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:PULS:LOW 0
ARB:PULS:LOW?
-> 0.000000000E+00
ARB:PULS:LOW 1,(@2)
ARB:PULS:LOW? (@1:2)
-> 0.000000000E+00,1.000000000E+00
```

Related commands

[SOURce:]ARB:PULSe:LOW:TIME, [SOURce:]ARB:PULSe:HIGH,
[SOURce:]ARB:FUNCtion:SHAPe, [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

13.18 [SOURce:]ARB:PULSe:LOW:TIME

Sets or queries how long the pulse shape of a channel holds its low level.

One execution begins with the low phase: the low level is held for this time, then the high level for ARB:PULSe:HIGH:TIME. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. A later change of ARB:TINTerval therefore changes how many DAC points the phase produces, not the pulse. The pause of ARB:SEQuence:PAUSe lies between executions and is not part of the low phase. The setting is per channel.

COMMAND

```
[SOURce:]ARB:PULSe:LOW:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Duration of the low phase. 0 s omits the phase.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:PULSe:LOW:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed low time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:PULS:LOW:TIME 2E-3
ARB:PULS:LOW:TIME?
-> 2.000000000E-03
ARB:PULS:LOW:TIME 1 ms,(@2)
ARB:PULS:LOW:TIME? (@1:2)
-> 2.000000000E-03,1.000000000E-03
```

Related commands

[SOURce:]ARB:PULSe:HIGH:TIME, [SOURce:]ARB:PULSe:LOW, [SOURce:]ARB:TINTerval, [SOURce:]ARB:SEQuence:PAUSe

13.19 [SOURce:]ARB:RAMP:END:TIME

Sets or queries how long the ramp shape of a channel holds its end level after the ramp.

One execution holds the start level for ARB:RAMP:STARt:TIME, ramps linearly to the end level over ARB:RAMP:RTIMe and finally holds the end level for this time; the length of one execution is the sum of the three times. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. Whether the end level stays on the output after the last execution is decided by ARB:TERMinate:LAST, not by this time. The setting is per channel.

COMMAND

```
[SOURce:]ARB:RAMP:END:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time at the end level. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:RAMP:END:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed hold time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:RAMP:END:TIME 0.2
ARB:RAMP:END:TIME?
-> 2.000000000E-01
ARB:RAMP:END:TIME 50 ms,(@2)
ARB:RAMP:END:TIME? (@1:2)
-> 2.000000000E-01,5.000000000E-02
```

Related commands

[SOURce:]ARB:RAMP:END[:LEVel], [SOURce:]ARB:RAMP:RTIMe, [SOURce:]ARB:RAMP:STARt:TIME, [SOURce:]ARB:TERMinate:LAST

13.20 [SOURce:]ARB:RAMP:END[:LEVel]

Sets or queries the end level of the ramp shape of a channel.

The level is the voltage setpoint the ramp reaches at the end of ARB:RAMP:RTIMe and then holds for ARB:RAMP:END:TIME, which closes one execution. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback. An end level below the start level is a falling ramp; the direction follows from the two levels, and the transition takes ARB:RAMP:RTIMe either way. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:RAMP:END[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<value>	NRf, V	Model-dependent	New voltage setpoint at the end of the ramp.
MINimum	CPD		Smallest programmable level (model-dependent).
MAXimum	CPD		Largest programmable level (model-dependent).
DEFault	CPD		Factory default, 0 V.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:RAMP:END[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V`	The programmed end level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:RAMP:END 10
ARB:RAMP:END?
-> 1.000000000E+01
ARB:RAMP:END 5,(@2)
ARB:RAMP:END? (@1:2)
-> 1.000000000E+01,5.000000000E+00
```

Related commands

[SOURce:]ARB:RAMP:STARt[:LEVel], [SOURce:]ARB:RAMP:RTIMe, [SOURce:]ARB:RAMP:END:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.21 [SOURce:]ARB:RAMP:RTIMe

Sets or queries the time the ramp shape of a channel takes from the start level to the end level.

One execution holds the start level for ARB:RAMP:STARt:TIME, ramps linearly from the start level to the end level over this time, then holds the end level for ARB:RAMP:END:TIME. The time is the duration of the transition itself and applies whether the end level lies above or below the start level. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The number of intermediate DAC steps of the ramp therefore follows from ARB:TINTerval, and a shorter update period resolves the same ramp more finely. The setting is per channel.

COMMAND

```
[SOURce:]ARB:RAMP:RTIMe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Duration of the transition from the start level to the end level. 0 s omits the phase.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:RAMP:RTIMe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed ramp time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:RAMP:RTIM 1
ARB:RAMP:RTIM?
-> 1.000000000E+00
ARB:RAMP:RTIM 250 ms,(@2)
ARB:RAMP:RTIM? (@1:2)
-> 1.000000000E+00,2.500000000E-01
```

Related commands

`[SOURce:]ARB:RAMP:STARt[:LEVel]`, `[SOURce:]ARB:RAMP:END[:LEVel]`, `[SOURce:]ARB:RAMP:STARt:TIME`, `[SOURce:]ARB:TINTerval`

13.22 [SOURce:]ARB:RAMP:STARt:TIME

Sets or queries how long the ramp shape of a channel holds its start level before the ramp.

One execution begins with this hold at the start level, then ramps to the end level over `ARB:RAMP:RTIMe` and holds the end level for `ARB:RAMP:END:TIME`. The hold is part of the waveform and is repeated with every execution, unlike the pause of `ARB:SEQuence:PAUSe`, which lies between executions. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel.

COMMAND

```
[SOURce:]ARB:RAMP:STARt:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Hold time at the start level before the ramp begins. 0 s omits the hold.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 0 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:RAMP:STARt:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 0 s, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s` The programmed hold time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	`"Data out of range"`	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:RAMP:STAR:TIME 0.1
ARB:RAMP:STAR:TIME?
-> 1.000000000E-01
ARB:RAMP:STAR:TIME 20 ms,(@2)
ARB:RAMP:STAR:TIME? (@1:2)
-> 1.000000000E-01,2.000000000E-02
```

Related commands

`[SOURce:]ARB:RAMP:STARt[:LEVel]`, `[SOURce:]ARB:RAMP:RTIMe`, `[SOURce:]ARB:RAMP:END:TIME`, `[SOURce:]ARB:SEQuence:PAUSe`

13.23 [SOURce:]ARB:RAMP:STARt[:LEVel]

Sets or queries the start level of the ramp shape of a channel.

The level is the voltage setpoint one execution starts at and holds for `ARB:RAMP:STARt:TIME` before the ramp to the end level begins. Native shapes modulate the voltage setpoint only; the current limit stays at its `[SOURce:]CURRent` value throughout playback. The level is in effect from the first DAC update point of the execution, and the ramp begins only after the start hold has elapsed. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:RAMP:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	New voltage setpoint at the start of the waveform.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:RAMP:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V`	The programmed start level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:RAMP:STAR 0
ARB:RAMP:STAR?
-> 0.000000000E+00
ARB:RAMP:STAR 1,(@2)
ARB:RAMP:STAR? (@1:2)
-> 0.000000000E+00,1.000000000E+00
```

Related commands

`[SOURce:]ARB:RAMP:END[:LEVel]`, `[SOURce:]ARB:RAMP:STARt:TIME`, `[SOURce:]ARB:RAMP:RTIMe`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.24 [SOURce:]ARB:SEQuence:COUNt

Sets or queries how many times a channel executes its programmed waveform.

One execution is the complete selected waveform: one generator cycle, or all segments of the user sequence from the first. The count applies in ARB:MODE SEQuence to every source, native shape, file-backed sequence and DATA sequence alike, and is a per-channel setting; it has no effect in STReam mode. INFinity repeats until ABORt ends the operation, and the query returns INF while it is selected. The pause of ARB:SEQuence:PAUSe is inserted between executions.

COMMAND

```
[SOURce:]ARB:SEQuence:COUNt
  <count>|INFinity[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<count>`	NR1	TBD	Number of executions of the complete waveform.
`INFinity`	CPD		Repeat until the operation is aborted.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same count is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SEQuence:COUNt?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR1> or <CRD>` The programmed number of executions, or `INF` while `INFinity` is selected.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The count is unchanged on every channel.
-222	`"Data out of range"`	`<count>` is outside the programmable range, or a listed channel does not exist. The count is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The count is unchanged.

Example

```
ARB:SEQ:COUN 5
ARB:SEQ:COUN?
-> 5
ARB:SEQ:COUN 10,(@2)
ARB:SEQ:COUN? (@1:2)
-> 5,10
```

Related commands

`[SOURce:]ARB:SEQuence:PAUSe`, `[SOURce:]ARB:TERMinate:LAST`, `[SOURce:]ARB:FUNCtion:SHAPe`, `ABORt`

13.25 [SOURce:]ARB:SEQuence:DATA

Loads or queries the in-memory user sequence of a channel as voltage, current, duration and interpolation quadruples.

One setter replaces the complete list of every addressed channel atomically; there is no append. The list is played when `ARB:FUNCtion:SHAPe SEQuence` and `ARB:SEQuence:SOURce DATA` are selected. Each segment holds its current limit as a step for its duration; `LINEAR` ramps the voltage setpoint from the previous segment's setpoint to its own over the duration and is therefore not permitted in the first segment. When playback starts the durations are sampled onto the `ARB:TINTerval` grid; the programmed values are kept. A file-backed `INITiate` loads its file into the same memory and overwrites this list; a failed load leaves the list empty. The query returns the list of exactly one channel, or `0` when that list is empty.

COMMAND

```
[SOURce:]ARB:SEQuence:DATA
  <voltage>,<current>,<duration>,<interpolation>[,<voltage>,<current>,<duration>,
  <interpolation>...][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<voltage>`	NRf, V	Model-dependent	Voltage setpoint of the segment.
`<current>`	NRf, A	Model-dependent	Current limit of the segment. Applied as a step at segment begin, never interpolated.
`<duration>`	NRf, s	10 us to 21 474.83647 s	Physical duration of the segment. Strictly positive; the shortest usable value is one tick of the 10 us timebase, the longest $2^{31}-1$ ticks.
`HOLD`	CPD		Hold the segment voltage setpoint for the duration.
`LINEAR`	CPD		Ramp the voltage setpoint linearly from the previous segment's setpoint to this one over the duration. Not permitted in the first segment.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same list is loaded into every listed channel. At most 4096 quadruples.

QUERY

```
[SOURce:]ARB:SEQuence:DATA?
  [(@<channel>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<channel>	Channel list		Exactly one channel to read. Omitted: channel 1.

Returns

<NR3>,<NR3>,<NR3>,<CRD> per segment	Voltage setpoint in V, current limit in A, duration in s and HOLD or LINEAR of every loaded segment, in list order. An empty list returns 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active, or the first segment is LINEAR. The previous list is unchanged.
-222	"Data out of range"	A voltage or current is outside the capability of the channel, a duration is zero, negative or longer than 21 474.83647 s, more than 4096 quadruples are given, or a listed channel does not exist. The previous list is unchanged.
-224	"Illegal parameter value"	The parameter count is not a whole number of quadruples or no parameter is given, or the query addresses more than one channel. The previous list is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The previous list is unchanged.

Example

```
ARB:SEQ:DATA 1,0.1,10E-6,HOLD,2,0.1,100E-6,LINEAR,3,0.2,1E-3,HOLD
ARB:SEQ:DATA?
-> 1.000000000E+00,1.000000000E-01,1.000000000E-05,HOLD,2.000000000E
   +00,1.000000000E-01,1.000000000E-04,LINEAR,3.000000000E+00,2.000000000E
   -01,1.000000000E-03,HOLD
ARB:SEQ:DATA:POIN?
-> 3
ARB:SEQ:DATA 5,1,1E-3,HOLD,(@1:2)
ARB:SEQ:DATA:POIN? (@1:2)
-> 1,1
```

Related commands

[SOURce:]ARB:SEQuence:DATA:POINts?, [SOURce:]ARB:SEQuence:SOURce, [SOURce:]ARB:SEQuence:FILE, [SOURce:]ARB:FUNCtion:SHAPe

13.26 [SOURce:]ARB:SEQuence:DATA:POINts?

Returns the number of segments loaded in the in-memory user sequence of a channel, or the segment limit.

The count reflects the last ARB:SEQuence:DATA setter, or the content of the sequence file loaded by the last successful INITiate with ARB:SEQuence:SOURce FILE, which overwrites the in-memory list. A channel without a loaded list reports 0, also after a failed file load, which leaves the list empty. MAXimum returns the common per-channel segment limit, which applies to DATA and file-backed sequences alike. Reading the count has no side effects.

QUERY

```
[SOURce:]ARB:SEQuence:DATA:POINts?
  [MAXimum][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MAXimum	CPD		Return the segment limit, 4096, instead of the loaded count.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The number of loaded segments, 0 without a loaded list, or 4096 with MAXimum.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
ARB:SEQ:DATA:POIN?
-> 3
ARB:SEQ:DATA:POIN? MAX
-> 4096
ARB:SEQ:DATA:POIN? (@1:2)
-> 3,0
```

Related commands

`[SOURce:]ARB:SEQuence:DATA`, `[SOURce:]ARB:SEQuence:FILE`, `[SOURce:]ARB:SEQuence:SOURce`

13.27 [SOURce:]ARB:SEQuence:FILE

Selects or queries the CSV file that provides the user sequence of a channel.

The setter stores the path only; the file is neither read nor checked at selection time. It is read once per start, during `INITiate` validation, when the channel participates with `ARB:FUNCtion:SHAPe SEQuence` and `ARB:SEQuence:SOURce FILE`, and is loaded completely into the channel's segment memory, overwriting an `ARB:SEQuence:DATA` list; playback never reads from storage. A missing or defective file, a first row with `LINEAR`, or more than 4096 data rows is reported at `INITiate` and arms nothing; a failed load leaves the channel's segment list empty, never partially loaded. The file is channel-local and holds one segment per row in the four columns `Voltage` in volts, `Current` in amperes, `Duration` in seconds with the limits of `ARB:SEQuence:DATA`, and `Interpolation`, either `HOLD` or `LINEAR`; the same file may be assigned to several channels. It is UTF-8 text, optionally with a byte order mark, with LF or CRLF line endings; blank lines and lines whose first non-blank character is # are ignored, so a column header is written as such a comment, and spaces and tabs around a field are ignored. An empty `Voltage`, `Current` or `Interpolation` field inherits the previous row, so the first data row has to fill every column. The file is read with the `SYSTem:CSV:DELimiter` and `SYSTem:CSV:DECimal` settings in effect at `INITiate`; selecting the path records no CSV setting.

COMMAND

```
[SOURce:]ARB:SEQuence:FILE
  "<path>"[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<path>`	String		Path of the CSV file; `USB:` prefix or unprefixed absolute path. An `INT:` path only on a model that offers that volume as a readable sequence source.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same file is assigned to every listed channel.

QUERY

```
[SOURce:]ARB:SEQuence:FILE?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<String> The configured path in quotes, or "" when no file is configured.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active; the path is unchanged on every channel. At INITiate, a participating channel selects SEQuence:SOURce FILE while no file is configured; nothing is armed.
-222	"Data out of range"	A listed channel does not exist; the path is unchanged. At INITiate, a row of the file has a range error, for example a duration below one DAC update period or longer than 21 474.83647 s; nothing is armed.
-223	"Too much data"	At INITiate, the file of a participating channel has more than 4096 data rows. Nothing is armed.
-171	"Invalid expression"	A channel appears twice in the channel list. The path is unchanged.
+310	"Sequence file syntax error"	At INITiate, a row of the file of a participating channel is malformed: wrong field count, a non-numeric field or an unknown shape keyword. Nothing is armed, and the segment list of that channel is left empty.

Example

```
ARB:SEQ:FILE "USB:/ramp.csv"
ARB:SEQ:FILE?
-> "USB:/ramp.csv"
ARB:SEQ:FILE "USB:/step.csv",(@2)
ARB:SEQ:FILE? (@1:2)
-> "USB:/ramp.csv","USB:/step.csv"
```

Related commands

[SOURce:]ARB:SEQuence:SOURce, [SOURce:]ARB:SEQuence:DATA, SYSTem:CSV:DELimiter, INITiate

13.28 [SOURce:]ARB:SEQuence:PAUSe

Sets or queries the pause a channel inserts between two executions of its programmed waveform.

The pause lies between complete executions counted by ARB:SEQuence:COUNt; it is not part of the waveform and is never inserted inside a generator cycle or between sequence segments. It applies in ARB:MODE SEQuence to every source and is a per-channel setting; it has no effect in STReam mode. INITiate rejects a participating channel whose single execution together with the pause lasts 0 s with -221. The query returns the programmed pause.

COMMAND

```
[SOURce:]ARB:SEQuence:PAUSe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	New pause between executions.
MINimum	CPD		Shortest programmable pause (TBD).
MAXimum	CPD		Longest programmable pause (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SEQuence:PAUSe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the shortest programmable pause instead of the setting.
MAXimum	CPD		Return the longest programmable pause instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed pause, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The pause is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The pause is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The pause is unchanged.

Example

```
ARB:SEQ:PAUS 0.5
ARB:SEQ:PAUS?
-> 5.000000000E-01
ARB:SEQ:PAUS? DEF
-> 0.000000000E+00
ARB:SEQ:PAUS 100 ms,(@2)
ARB:SEQ:PAUS? (@1:2)
-> 5.000000000E-01,1.000000000E-01
```

Related commands

[SOURce:]ARB:SEQuence:COUNt, [SOURce:]ARB:TERMinate:LAST, [SOURce:]ARB:FUNCtion:SHAPe

13.29 [SOURce:]ARB:SEQuence:SOURce

Selects or queries whether the user sequence of a channel comes from a CSV file or from the in-memory list.

The selection takes effect with ARB:FUNCtion:SHAPe SEQuence. FILE plays the file configured with ARB:SEQuence:FILE, which is read completely at INITiate. DATA plays the list loaded with ARB:SEQuence:DATA. Both use the same segment model and the same 4096-segment memory of the channel, so after a successful file load DATA plays the loaded file content until a new list is sent, and after a failed load the list is empty. A participating channel with FILE and no configured file, or with DATA and an empty list, fails INITiate validation with -221 and arms nothing.

COMMAND

```
[SOURce:]ARB:SEQuence:SOURce
  FILE|DATA[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
FILE	CPD		CSV file selected with ARB:SEQuence:FILE.
DATA	CPD		In-memory list loaded with ARB:SEQuence:DATA.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same selection is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SEQuence:SOURce?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<CRD> FILE or DATA.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active; the selection is unchanged on every channel. At INITiate, a participating channel selects FILE while no file is configured, or DATA while its list is empty; nothing is armed.
-222	"Data out of range"	A listed channel does not exist. The selection is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The selection is unchanged.

Example

```
ARB:FUNC:SHAP SEQ
ARB:SEQ:SOUR DATA
ARB:SEQ:SOUR?
-> DATA
ARB:SEQ:SOUR FILE,(@2)
ARB:SEQ:SOUR? (@1:2)
-> DATA,FILE
```

Related commands

`[SOURce:]ARB:FUNCtion:SHAPe`, `[SOURce:]ARB:SEQuence:FILE`, `[SOURce:]ARB:SEQuence:DATA`

13.30 [SOURce:]ARB:SINusoid:AMPLitude

Sets or queries the peak amplitude of the sinusoid shape.

One execution of the shape is exactly one sine period, and the amplitude is its peak deviation from `ARB:SINusoid:OFFSet`, so the setpoint swings between offset minus amplitude and offset plus amplitude. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its `[SOURce:]CURRent` value, and the regulator enters CC as it would for a static setpoint. Both extremes have to lie within the voltage capability of the channel; the setters do not check amplitude and offset against each other, and `INITiate` rejects a participating channel whose sine would leave the capability. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe SINusoid` in `ARB:MODE SEQuence`. The query returns the programmed amplitude, not a measurement.

COMMAND

```
[SOURce:]ARB:SINusoid:AMPLitude
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Peak amplitude of the sine around the offset.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SINusoid:AMPLitude?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed peak amplitude of the sinusoid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active; the amplitude is unchanged on every channel. At `INITiate`, the sine of a participating `SINusoid` channel would leave the voltage capability of the model, for example with an offset smaller than the amplitude; nothing is armed.
-222	`"Data out of range"`	`<value>` is outside the capability of the channel, or a listed channel does not exist. The amplitude is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The amplitude is unchanged.

Example

```
ARB:SIN:AMPL 2
ARB:SIN:AMPL?
-> 2.000000000E+00
ARB:SIN:AMPL 0.5,(@2)
ARB:SIN:AMPL? (@1:2)
-> 2.000000000E+00,5.000000000E-01
```

Related commands

`[SOURce:]ARB:SINusoid:OFFSet`, `[SOURce:]ARB:SINusoid:FREQuency`, `[SOURce:]ARB:SINusoid:PHASe`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.31 [SOURce:]ARB:SINusoid:FREQuency

Sets or queries the frequency of the sinusoid shape.

One execution of the shape is exactly one sine period, so the frequency defines the length of an execution as its reciprocal; periodic operation is built from it with ARB:SEQuence:COUNt and, where a gap is wanted, ARB:SEQuence:PAUSe. When playback starts the resulting period is sampled onto the ARB:TINTerval grid; the programmed frequency is kept. A high frequency combined with a long update period therefore yields few DAC points per period and a visibly stepped sine. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe SINusoid in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:SINusoid:FREQuency
  <hertz>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<hertz>	NRf, Hz	TBD	Frequency of the sine.
MINimum	CPD		Smallest programmable frequency (TBD).
MAXimum	CPD		Largest programmable frequency (TBD).
DEFault	CPD		Factory default, 1 kHz.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SINusoid:FREQuency?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable frequency instead of the setting.
MAXimum	CPD		Return the largest programmable frequency instead of the setting.
DEFault	CPD		Return the factory default, 1 kHz, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, Hz	The programmed frequency of the sinusoid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The frequency is unchanged on every channel.
-222	"Data out of range"	<hertz> is outside the programmable range, or a listed channel does not exist. The frequency is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The frequency is unchanged.

Example

```
ARB:SIN:FREQ 50
ARB:SIN:FREQ?
-> 5.000000000E+01
ARB:SIN:FREQ 1 kHz,(@2)
ARB:SIN:FREQ? (@1:2)
-> 5.000000000E+01,1.000000000E+03
```

Related commands

[SOURce:]ARB:SINusoid:AMPLitude, [SOURce:]ARB:SINusoid:PHASe, [SOURce:]ARB:TINTerval, [SOURce:]ARB:SEQuence:COUNt

13.32 [SOURce:]ARB:SINusoid:OFFSet

Sets or queries the DC offset the sinusoid shape swings around.

The offset is the centre of the one sine period that makes up one execution: the setpoint moves between offset minus ARB:SINusoid:AMPLitude and offset plus amplitude, and equals the offset itself at phase 0 and 180 degrees. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its [SOURce:]CURRent value. Offset and amplitude have to be chosen so that both extremes stay within the capability of the channel, which INITiate checks for every participating channel; the offset alone does not shift the present output, which follows [SOURce:]VOLTage until playback begins. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe SINusoid in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:SINusoid:OFFSet
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<value>	NRf, V	Model-dependent	DC voltage setpoint the sine is centred on.
MINimum	CPD		Smallest programmable level (model-dependent).
MAXimum	CPD		Largest programmable level (model-dependent).
DEFault	CPD		Factory default, 0 V.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SINusoid:OFFSet?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The programmed DC offset of the sinusoid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active; the offset is unchanged on every channel. At INITiate, the sine of a participating SINusoid channel would leave the voltage capability of the model, for example with an offset smaller than the amplitude; nothing is armed.
-222	"Data out of range"	<value> is outside the capability of the channel, or a listed channel does not exist. The offset is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The offset is unchanged.

Example

```
ARB:SIN:OFFS 5
ARB:SIN:OFFS?
-> 5.000000000E+00
ARB:SIN:OFFS 3,(@2)
ARB:SIN:OFFS? (@1:2)
-> 5.000000000E+00,3.000000000E+00
```

Related commands

`[SOURce:]ARB:SINusoid:AMPLitude`, `[SOURce:]ARB:SINusoid:FREQuency`, `[SOURce:]ARB:SINusoid:PHASe`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.33 [SOURce:]ARB:SINusoid:PHASe

Sets or queries the phase at which one execution of the sinusoid shape starts.

One execution is exactly one sine period, and the phase selects the point of that period at which the execution begins and, one full period later, ends. With phase 0 the execution starts at the offset level and rises first; with 90 degrees it starts at the positive peak. Every repetition starts at the same phase, and because one execution is a full period the setpoint at the end of an execution equals the setpoint at its start, so consecutive executions join without a step at any phase. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe SINusoid` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:SINusoid:PHASe
  <degrees>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<degrees>`	NRf, deg	0 to 360	Phase at which one execution of the sine starts.
`MINimum`	CPD		Smallest programmable phase, 0 degrees.
`MAXimum`	CPD		Largest programmable phase, 360 degrees.
`DEFault`	CPD		Factory default, 0 degrees.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:SINusoid:PHASe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable phase instead of the setting.
`MAXimum`	CPD		Return the largest programmable phase instead of the setting.
`DEFault`	CPD		Return the factory default, 0 degrees, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, deg` The programmed start phase of the sinusoid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The phase is unchanged on every channel.
-222	`"Data out of range"`	`<degrees>` is outside 0 to 360, or a listed channel does not exist. The phase is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The phase is unchanged.

Example

```
ARB:SIN:PHAS 90
ARB:SIN:PHAS?
-> 9.000000000E+01
ARB:SIN:PHAS 0,(@2)
ARB:SIN:PHAS? (@1:2)
-> 9.000000000E+01,0.000000000E+00
```

Related commands

`[SOURce:]ARB:SINusoid:FREQuency`, `[SOURce:]ARB:SINusoid:AMPLitude`, `[SOURce:]ARB:SINusoid:OFFSet`, `[SOURce:]ARB:SEQuence:COUNt`

13.34 [SOURce:]ARB:STAircase:END:TIME

Sets or queries how long the linear staircase holds its end level after the last step.

One execution of the shape is: hold the start level for STARt:TIME, run NSTeps equally spaced levels over TIME, then hold the end level for this time. The last step is already exactly the end level and is counted inside the staircase time, so this hold extends the end level rather than producing it. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe STAircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:STAircase:END:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time of the end level after the staircase has completed. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:END:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed hold time of the end level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The hold time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:STA:END:TIME 0.2
ARB:STA:END:TIME?
-> 2.000000000E-01
ARB:STA:END:TIME 1,(@2)
ARB:STA:END:TIME? (@1:2)
-> 2.000000000E-01,1.000000000E+00
```

Related commands

[SOURce:]ARB:STAircase:END[:LEVel], [SOURce:]ARB:STAircase:TIME, [SOURce:]ARB:STAircase:STARt:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.35 [SOURce:]ARB:STAircase:END[:LEVel]

Sets or queries the level the last step of the linear staircase reaches.

For step k = 1...NSTeps the level is STARt + k * (END - STARt) / NSTeps, so the last step is exactly this end level and lies inside ARB:STAircase:TIME; ARB:STAircase:END:TIME holds it afterwards. The end level may lie below the start level, which makes the staircase descend; the step size is the difference divided by ARB:STAircase:NSTeps. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its [SOURce:]CURRent value. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe STAircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:STAircase:END[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint of the last staircase step.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:END[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed end level of the linear staircase, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The end level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the capability of the channel, or a listed channel does not exist. The end level is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The end level is unchanged.

Example

```
ARB:STA:END 10
ARB:STA:END?
-> 1.000000000E+01
ARB:STA:END 4,(@2)
ARB:STA:END? (@1:2)
-> 1.000000000E+01,4.000000000E+00
```

Related commands

`[SOURce:]ARB:STAircase:STARt[:LEVel]`, `[SOURce:]ARB:STAircase:NSTeps`, `[SOURce:]ARB:STAircase:END:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.36 [SOURce:]ARB:STAircase:NSTeps

Sets or queries how many equally spaced steps the linear staircase executes.

The steps are distributed over `ARB:STAircase:TIME`, each one lasting `TIME / NSTeps`, and step k has the level `STARt + k * (END - STARt) / NSTeps`, so the count sets both the step height and the dwell of a single step. The start level is not itself one of the steps: the first step lies between the start and the end level, and the last step is exactly `ARB:STAircase:END`. At least one step is required, and increasing the count makes the steps smaller and shorter without changing the total staircase time. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe STAircase` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:STAircase:NSTeps
  <count>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<count>`	NR1	1 to TBD	Number of equally spaced staircase steps in one execution.
`MINimum`	CPD		Smallest programmable step count (TBD).
`MAXimum`	CPD		Largest programmable step count (TBD).
`DEFault`	CPD		Factory default, 10.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:NSTeps?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable step count instead of the setting.
MAXimum	CPD		Return the largest programmable step count instead of the setting.
DEFault	CPD		Return the factory default, 10, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> The programmed number of staircase steps, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The step count is unchanged on every channel.
-222	"Data out of range"	<count> is smaller than 1 or above the programmable maximum, or a listed channel does not exist. The step count is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The step count is unchanged.

Example

```
ARB:STA:NST 10
ARB:STA:NST?
-> 10
ARB:STA:NST 5,(@2)
ARB:STA:NST? (@1:2)
-> 10,5
```

Related commands

[SOURce:]ARB:STAircase:TIME, [SOURce:]ARB:STAircase:STARt[:LEVel], [SOURce:]ARB:STAircase:END[:LEVel], [SOURce:]ARB:FUNCtion:SHAPe

13.37 [SOURce:]ARB:STAircase:STARt:TIME

Sets or queries how long the linear staircase holds its start level before the first step.

One execution begins with this hold of ARB:STAircase:STARt; the ARB:STAircase:NSTeps steps over ARB:STAircase:TIME follow, and ARB:STAircase:END:TIME closes the execution. The hold is part of the waveform and occurs in every execution, unlike ARB:SEQuence:PAUSe, which lies between executions. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe STAircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:STAircase:STARt:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time of the start level before the first step. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:STARt:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed hold time of the start level, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The hold time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The hold time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The hold time is unchanged.

Example

```
ARB:STA:STAR:TIME 0.1
ARB:STA:STAR:TIME?
-> 1.000000000E-01
ARB:STA:STAR:TIME 20 ms,(@2)
ARB:STA:STAR:TIME? (@1:2)
-> 1.000000000E-01,2.000000000E-02
```

Related commands

[SOURce:]ARB:STAircase:STARt[:LEVel], [SOURce:]ARB:STAircase:TIME, [SOURce:]ARB:STAircase:END:TIME, [SOURce:]ARB:FUNCtion:SHAPe

13.38 [SOURce:]ARB:STAircase:STARt[:LEVel]

Sets or queries the level the linear staircase starts from.

The level is held for ARB:STAircase:STARt:TIME at the beginning of one execution and is the base of the progression STARt + k * (END - STARt) / NSTeps, so it is not itself one of the ARB:STAircase:NSTeps steps; the first step already lies one increment above or below it. The value programs the voltage setpoint in volts; native shapes do not modulate the current limit, which stays at its [SOURce:]CURRent value. Programming the level neither starts playback nor changes the present output, which follows [SOURce:]VOLTage until playback begins. The setting is per channel and takes effect with ARB:FUNCtion:SHAPe STAircase in ARB:MODE SEQuence.

COMMAND

```
[SOURce:]ARB:STAircase:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	Voltage setpoint at the beginning of one execution.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed start level of the linear staircase, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The start level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the capability of the channel, or a listed channel does not exist. The start level is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The start level is unchanged.

Example

```
ARB:STA:STAR 0
ARB:STA:STAR?
-> 0.000000000E+00
ARB:STA:STAR 1,(@2)
ARB:STA:STAR? (@1:2)
-> 0.000000000E+00,1.000000000E+00
```

Related commands

`[SOURce:]ARB:STAircase:END[:LEVel]`, `[SOURce:]ARB:STAircase:NSTeps`, `[SOURce:]ARB:STAircase:STARt:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.39 [SOURce:]ARB:STAircase:TIME

Sets or queries the total time the linear staircase needs for all its steps.

The time covers the complete staircase from the first step to the last and is divided equally, so one step lasts `TIME / NSTeps`; it excludes the start hold of `ARB:STAircase:STARt:TIME` and the end hold of `ARB:STAircase:END:TIME`, and one execution therefore takes the sum of the three. Because the last step is already `ARB:STAircase:END`, the end level is reached at the end of this time, not after it. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel and takes effect with `ARB:FUNCtion:SHAPe STAircase` in `ARB:MODE SEQuence`.

COMMAND

```
[SOURce:]ARB:STAircase:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Total time of all staircase steps. 0 s omits the phase.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 1 ms.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:STAircase:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 1 ms, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s` The programmed total staircase time, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The staircase time is unchanged on every channel.
-222	`"Data out of range"`	`<seconds>` is outside the programmable range, or a listed channel does not exist. The staircase time is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The staircase time is unchanged.

Example

```
ARB:STA:TIME 1
ARB:STA:TIME?
-> 1.000000000E+00
ARB:STA:TIME 0.5,(@2)
ARB:STA:TIME? (@1:2)
-> 1.000000000E+00,5.000000000E-01
```

Related commands

`[SOURce:]ARB:STAircase:NSTeps`, `[SOURce:]ARB:STAircase:STARt:TIME`, `[SOURce:]ARB:STAircase:END:TIME`, `[SOURce:]ARB:TINTerval`

13.40 [SOURce:]ARB:STReam:DATA

Appends one block of voltage setpoint and current limit points to the host-fed DAC stream.

The block is queued in playback order behind the points already accepted; every point is output for one ARB:TINTerval. A point carries one voltage setpoint followed by one current limit for every channel with ARB[:STATe] ON, in ascending channel order, without placeholders for other channels. STReam:DATA, STReam:DATA:VOLTage and STReam:DATA:CURRent feed the same stream and may be mixed block by block. The command is accepted only in ARB:MODE STReam with a configured STReam:POINts and is atomic: a rejected block changes neither the buffer nor the last-valid values, and a block that breaks several rules reports only the first failing check in the order payload size, mode, participating channels, configured length, block form, room and values. Blocks are preloaded before INITiate, which keeps the buffer and requires at least one complete block, or the whole stream when it is shorter, and appended after t0 while older points are consumed; DAC_BUFFER_FREE in the Measurement status reports that one complete block can be accepted immediately. A stream that runs empty before STReam:POINts points were produced ends with a DAC underrun, which ends every participating channel abnormally and restores the pre-playback setpoints and limits.

COMMAND

```
[SOURce:]ARB:STReam:DATA
  <binary-block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<binary-block>	Block		Definite-length arbitrary block #<n><byte count><payload>. The payload is 4096 points of little-endian IEEE-754 binary32 values, independent of FORMat:BORDer and FORMat[:DATA], per point V, I per participating channel in ascending channel order, in volts and amperes; 8 bytes per point and channel. Only the final block may be shorter, and only when it brings the accepted total exactly to STReam:POINts.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	ARB:MODE is SEQuence, no channel is ARB[:STATe] ON, STReam:POINts is 0 (not configured), or the buffer has no room for a complete block. Nothing is queued.
-222	"Data out of range"	A value is not a finite number or lies outside the voltage or current capability. Nothing is queued.
-161	"Invalid block data"	The byte count is not a whole number of points or exceeds the largest block, the block does not hold 4096 points and is not the exact final block, or the block would exceed STReam:POINts, as every block after the complete stream does. Nothing is queued. The indefinite-length form #0 is not accepted either. The indefinite-length form #0 is not accepted either.

Example

```
ARB:MODE STR
ARB:STR:POIN 8192
ARB:STR:DATA #532768<32768 bytes>
ARB:STR:DATA #532768<32768 bytes>
OUTP ON
INIT TRAN
```

Related commands

`[SOURce:]ARB:STReam:POINts`, `[SOURce:]ARB:STReam:DATA:VOLTage`, `[SOURce:]ARB:STReam:DATA:CURRent`, `STATus:MEASurement:CONDition?`

13.41 [SOURce:]ARB:STReam:DATA:CURRent

Appends one block of current limit points to the host-fed DAC stream.

The block behaves like `ARB:STReam:DATA` with one value per point and participating channel. The voltage setpoint of every point in the block is the last valid voltage of that channel: the voltage of the last point queued since the buffer was last cleared, or, when none was queued since, the channel's `[SOURce:]VOLTage` setpoint at the moment the block is accepted. Points already queued never change afterwards. The same acceptance, buffer and underrun rules as for `ARB:STReam:DATA` apply.

COMMAND

```
[SOURce:]ARB:STReam:DATA:CURRent
  <binary-block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<binary-block>`	Block		Definite-length arbitrary block `#<n><byte count><payload>`. The payload is 4096 points of little-endian IEEE-754 binary32 current limits, independent of `FORMat:BORDer` and `FORMat[:DATA]`, per point one value per participating channel in ascending channel order, in amperes; 4 bytes per point and channel. Only the final block may be shorter, and only when it brings the accepted total exactly to `STReam:POINts`.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	ARB:MODE is SEQuence, no channel is ARB[:STATe] ON, STReam:POINts is 0 (not configured), or the buffer has no room for a complete block. Nothing is queued.
-222	"Data out of range"	A value is not a finite number or lies outside the current capability. Nothing is queued.
-161	"Invalid block data"	The byte count is not a whole number of points or exceeds the largest block, the block does not hold 4096 points and is not the exact final block, or the block would exceed STReam:POINts, as every block after the complete stream does. Nothing is queued. The indefinite-length form #0 is not accepted either. The indefinite-length form #0 is not accepted either.

Example

```
ARB:MODE STR
ARB:STR:POIN 4096
ARB:STR:DATA:CURR #516384<16384 bytes>
OUTP ON
INIT TRAN
```

Related commands

[SOURce:]ARB:STReam:DATA, [SOURce:]ARB:STReam:DATA:VOLTage, [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude], [SOURce:]ARB:STReam:POINts

13.42 [SOURce:]ARB:STReam:DATA:VOLTage

Appends one block of voltage setpoint points to the host-fed DAC stream.

The block behaves like ARB:STReam:DATA with one value per point and participating channel. The current limit of every point in the block is the last valid current of that channel: the limit of the last point queued since the buffer was last cleared, or, when none was queued since, the channel's [SOURce:]CURRent limit at the moment the block is accepted. Points already queued never change afterwards. The same acceptance, buffer and underrun rules as for ARB:STReam:DATA apply.

COMMAND

```
[SOURce:]ARB:STReam:DATA:VOLTage
  <binary-block>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<binary-block>	Block		Definite-length arbitrary block #<n><byte count><payload>. The payload is 4096 points of little-endian IEEE-754 binary32 voltage setpoints, independent of FORMat:BORDer and FORMat[:DATA], per point one value per participating channel in ascending channel order, in volts; 4 bytes per point and channel. Only the final block may be shorter, and only when it brings the accepted total exactly to STReam:POINts.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	ARB:MODE is SEQuence, no channel is ARB[:STATe] ON, STReam:POINts is 0 (not configured), or the buffer has no room for a complete block. Nothing is queued.
-222	"Data out of range"	A value is not a finite number or lies outside the voltage capability. Nothing is queued.
-161	"Invalid block data"	The byte count is not a whole number of points or exceeds the largest block, the block does not hold 4096 points and is not the exact final block, or the block would exceed STReam:POINts, as every block after the complete stream does. Nothing is queued. The indefinite-length form #0 is not accepted either. The indefinite-length form #0 is not accepted either.

Example

```
ARB:MODE STR
ARB:STR:POIN 4096
ARB:STR:DATA:VOLT #516384<16384 bytes>
OUTP ON
INIT TRAN
```

Related commands

[SOURce:]ARB:STReam:DATA, [SOURce:]ARB:STReam:DATA:CURRent, [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude], [SOURce:]ARB:STReam:POINts

13.43 [SOURce:]ARB:STReam:POINts

Sets or queries the total number of DAC update points of the host-fed stream.

The count is global and finite: the stream ends normally when this many points have been output on all participating channels together, and its duration is the count times ARB:TINTerval. It bounds the data accepted by the STReam:DATA commands and decides which block may be shorter than 4096 points. The count may be raised while points are buffered; a count below the number of points already accepted cuts the buffered stream to that length, discards the points beyond it and takes the last-valid values from the new last point. The setting has no effect in ARB:MODE SEQuence. The query returns the programmed total, not the number of points accepted or played.

COMMAND

```
[SOURce:]ARB:STReam:POINts
  <count>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<count>	NR1	0 to TBD	Total number of stream points to output. 0 means that no stream is configured; the STReam:DATA commands and INITiate in STReam mode then generate -221.

QUERY

```
[SOURce:]ARB:STReam:POINts?
```

Returns

<NR1>	The programmed total point count.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The count and the buffer are unchanged.
-222	"Data out of range"	<count> is outside the programmable range. The count is unchanged.

Example

```
ARB:STR:POIN 100000
ARB:STR:POIN?
-> 100000
```

Related commands

`[SOURce:]ARB:STReam:DATA`, `[SOURce:]ARB:TINTerval`, `[SOURce:]ARB[:STATe]`, `[SOURce:]ARB:MODE`

13.44 [SOURce:]ARB:TERMinate:LAST

Selects or queries whether a channel holds the final waveform value or restores its setpoints after playback completes normally.

The setting applies to every playback mode and source and is a per-channel setting. With `ON` the voltage setpoint, and for a user sequence or the stream also the current limit, stays at the last value the waveform reached; with `OFF` the values programmed before playback started are restored. Normal completion means the waveform ran to its configured end without abort or fault. Every abnormal end, for example `ABORt`, a DAC underrun, `OUTPut OFF` or a protection trip on a participating channel, or an active or latched inhibit input, ends the playback of every participating channel and restores the pre-playback values regardless of this setting. A channel that finishes before the other participants is held at its termination value until the whole DAC participant has ended.

COMMAND

```
[SOURce:]ARB:TERMinate:LAST
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ON | 1`	Boolean		Hold the final waveform value after normal completion.
`OFF | 0`	Boolean		Restore the pre-playback voltage setpoint and current limit after normal completion.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TERMinate:LAST?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1>	1 when the final value is held, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The state is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The state is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The state is unchanged.

Example

```
ARB:TERM:LAST ON
ARB:TERM:LAST?
-> 1
ARB:TERM:LAST OFF,(@2)
ARB:TERM:LAST? (@1:2)
-> 1,0
```

Related commands

ABORt, [SOURce:]ARB:SEQuence:COUNt,
[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude],
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

13.45 [SOURce:]ARB:TINTerval

Sets or queries the common DAC update period of every ARB source.

The period is global and is the time between two consecutive DAC updates for native shapes, user sequences and the host-fed stream alike. It lies on the 10 us DAC grid: a requested value is quantized to the nearest multiple of 10 us, with exact halfway cases rounded to the larger period, and the query returns the quantized value. The period selects the sampling grid only. Shape times and segment durations keep their programmed values in seconds and are sampled onto the grid when playback starts, so changing the period changes how many DAC points a waveform produces, not the waveform; a time boundary between two update points moves to the nearest one, an exact halfway case to the later one. A stream point, in contrast, is output for exactly one period.

COMMAND

```
[SOURce:]ARB:TINTerval
  <seconds>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	10 us to MAXimum (TBD)	New update period. Quantized to the nearest multiple of 10 us.
MINimum	CPD		One DAC grid tick, 10 us, the maximum update rate of 100 kSa/s.
MAXimum	CPD		Longest programmable period (TBD).
DEFault	CPD		Factory default (TBD).

QUERY

```
[SOURce:]ARB:TINTerval?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the shortest programmable period, 10 us, instead of the setting.
MAXimum	CPD		Return the longest programmable period instead of the setting.
DEFault	CPD		Return the factory default instead of the setting.

Returns

<NR3>, s — The configured, quantized update period, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The period is unchanged.
-222	"Data out of range"	<seconds> is outside MINimum to MAXimum. The period is unchanged.

Example

```
ARB:TINT 100E-6
ARB:TINT?
-> 1.000000000E-04
ARB:TINT 25 us
ARB:TINT?
-> 3.000000000E-05
ARB:TINT? MIN
-> 1.000000000E-05
```

Related commands

`[SOURce:]ARB:STReam:POINts`, `[SOURce:]ARB:SEQuence:DATA`, `[SOURce:]ARB:FUNCtion:SHAPe`, `[SOURce:]ARB:MODE`

13.46 [SOURce:]ARB:TRAPezoid:END:TIME

Sets or queries how long the trapezoid shape of a channel holds the start level after the falling edge.

One execution holds the start level for `ARB:TRAPezoid:STARt:TIME`, rises to the top level over `ARB:TRAPezoid:RTIMe`, holds the top level for `ARB:TRAPezoid:TOP:TIME`, falls back to the start level over `ARB:TRAPezoid:FTIMe` and holds the start level for this time. The trapezoid deliberately has no separate end level: the waveform returns to `ARB:TRAPezoid:STARt[:LEVel]`, so this is a second hold at the start level and closes one execution. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRAPezoid:END:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Hold time at the start level after the falling edge. 0 s omits the hold.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 0 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:END:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.

`DEFault`	CPD	Return the factory default, 0 s, instead of the setting.
`<chanlist>`	Channel list	Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s`	The programmed hold time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRAP:END:TIME 0.1
ARB:TRAP:END:TIME?
-> 1.000000000E-01
ARB:TRAP:END:TIME 25 ms,(@2)
ARB:TRAP:END:TIME? (@1:2)
-> 1.000000000E-01,2.500000000E-02
```

Related commands

`[SOURce:]ARB:TRAPezoid:STARt[:LEVel]`, `[SOURce:]ARB:TRAPezoid:FTIMe`, `[SOURce:]ARB:TRAPezoid:STARt:TIME`, `[SOURce:]ARB:TERMinate:LAST`

13.47 [SOURce:]ARB:TRAPezoid:FTIMe

Sets or queries the fall time of the trapezoid shape of a channel.

The fall is the fourth phase of one execution: after the top hold the setpoint returns linearly from the top level to the start level over this time, and the execution then holds the start level for `END:TIME`. Rise and fall are programmed independently, so an asymmetric trapezoid needs no sequence. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The number of intermediate DAC steps of the edge therefore follows from `ARB:TINTerval`. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRAPezoid:FTIMe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Duration of the transition from the top level back to the start level. 0 s omits the phase.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 1 ms.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:FTIMe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 1 ms, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s` — The programmed fall time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The time is unchanged on every channel.
-222	`"Data out of range"`	`<seconds>` is outside the programmable range, or a listed channel does not exist. The time is unchanged.

-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRAP:FTIM 5E-3
ARB:TRAP:FTIM?
-> 5.000000000E-03
ARB:TRAP:FTIM 2 ms,(@2)
ARB:TRAP:FTIM? (@1:2)
-> 5.000000000E-03,2.000000000E-03
```

Related commands

`[SOURce:]ARB:TRAPezoid:RTIMe`, `[SOURce:]ARB:TRAPezoid:TOP:TIME`, `[SOURce:]ARB:TRAPezoid:END:TIME`, `[SOURce:]ARB:TINTerval`

13.48 [SOURce:]ARB:TRAPezoid:RTIMe

Sets or queries the rise time of the trapezoid shape of a channel.

The rise is the second phase of one execution: after the start hold the setpoint moves linearly from the start level to the top level over this time, and the top level is then held for `ARB:TRAPezoid:TOP:TIME`. The time is the duration of the transition itself and applies whether the top level lies above or below the start level. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The number of intermediate DAC steps of the edge therefore follows from `ARB:TINTerval`. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRAPezoid:RTIMe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Duration of the transition from the start level to the top level. 0 s omits the phase.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 1 ms.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:RTIMe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed rise time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRAP:RTIM 5E-3
ARB:TRAP:RTIM?
-> 5.000000000E-03
ARB:TRAP:RTIM 10 ms,(@2)
ARB:TRAP:RTIM? (@1:2)
-> 5.000000000E-03,1.000000000E-02
```

Related commands

[SOURce:]ARB:TRAPezoid:FTIMe, [SOURce:]ARB:TRAPezoid:TOP[:LEVel], [SOURce:]ARB:TRAPezoid:STARt:TIME, [SOURce:]ARB:TINTerval

13.49 [SOURce:]ARB:TRAPezoid:STARt:TIME

Sets or queries how long the trapezoid shape of a channel holds its start level before the rising edge.

One execution begins with this hold at the start level, then rises to the top level over RTIMe. It is the first of the two holds at the start level; the second, after the falling edge, is END:TIME, and the two are programmed independently. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRAPezoid:STARt:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Hold time at the start level before the rising edge. 0 s omits the hold.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 0 s.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:STARt:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 0 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed hold time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRAP:STAR:TIME 0.05
ARB:TRAP:STAR:TIME?
-> 5.000000000E-02
ARB:TRAP:STAR:TIME 10 ms,(@2)
ARB:TRAP:STAR:TIME? (@1:2)
-> 5.000000000E-02,1.000000000E-02
```

Related commands

[SOURce:]ARB:TRAPezoid:STARt[:LEVel], [SOURce:]ARB:TRAPezoid:RTIMe, [SOURce:]ARB:TRAPezoid:END:TIME, [SOURce:]ARB:SEQuence:PAUSe

13.50 [SOURce:]ARB:TRAPezoid:STARt[:LEVel]

Sets or queries the start level of the trapezoid shape of a channel.

The level is the voltage setpoint one execution starts at, held for ARB:TRAPezoid:STARt:TIME, and it is also the level the falling edge returns to and holds for ARB:TRAPezoid:END:TIME; the trapezoid has no separate end level. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback. Because both holds use this one level, a repeated trapezoid is continuous: the last value of one execution is the first value of the next. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:TRAPezoid:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<value>	NRf, V	Model-dependent	New voltage setpoint of the base level, used before the rising edge and after the falling edge.
MINimum	CPD		Smallest programmable level (model-dependent).

`MAXimum`	CPD	Largest programmable level (model-dependent).
`DEFault`	CPD	Factory default, 0 V.
`<chanlist>`	Channel list	Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed start level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:TRAP:STAR 0
ARB:TRAP:STAR?
-> 0.000000000E+00
ARB:TRAP:STAR 2,(@2)
ARB:TRAP:STAR? (@1:2)
-> 0.000000000E+00,2.000000000E+00
```

Related commands

`[SOURce:]ARB:TRAPezoid:TOP[:LEVel]`, `[SOURce:]ARB:TRAPezoid:STARt:TIME`, `[SOURce:]ARB:TRAPezoid:END:TIME`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.51 [SOURce:]ARB:TRAPezoid:TOP:TIME

Sets or queries how long the trapezoid shape of a channel holds its top level.

The top hold is the third phase of one execution: it follows the rising edge of `ARB:TRAPezoid:RTIMe` and is followed by the falling edge of `ARB:TRAPezoid:FTIMe`. The length of one execution is the sum of the two holds at the start level, this hold and the two edge times. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRAPezoid:TOP:TIME
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Hold time at the top level between the rising and the falling edge. 0 s omits the hold.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 0 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:TOP:TIME?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable time instead of the setting.
`MAXimum`	CPD		Return the largest programmable time instead of the setting.
`DEFault`	CPD		Return the factory default, 0 s, instead of the setting.

<chanlist>	Channel list	Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed hold time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRAP:TOP:TIME 0.5
ARB:TRAP:TOP:TIME?
-> 5.000000000E-01
ARB:TRAP:TOP:TIME 100 ms,(@2)
ARB:TRAP:TOP:TIME? (@1:2)
-> 5.000000000E-01,1.000000000E-01
```

Related commands

[SOURce:]ARB:TRAPezoid:TOP[:LEVel], [SOURce:]ARB:TRAPezoid:RTIMe, [SOURce:]ARB:TRAPezoid:FTIMe, [SOURce:]ARB:TINTerval

13.52 [SOURce:]ARB:TRAPezoid:TOP[:LEVel]

Sets or queries the top level of the trapezoid shape of a channel.

The level is the voltage setpoint the rising edge reaches at the end of ARB:TRAPezoid:RTIMe and holds for ARB:TRAPezoid:TOP:TIME before the falling edge returns to the start level. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback. A top level below the start level inverts the trapezoid; the direction follows from the two levels, and the edges keep their programmed times either way. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:TRAPezoid:TOP[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	New voltage setpoint of the top plateau.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRAPezoid:TOP[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` — The programmed top level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the programmable range, or a listed channel does not exist. The level is unchanged.

-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:TRAP:TOP 12
ARB:TRAP:TOP?
-> 1.200000000E+01
ARB:TRAP:TOP 5,(@2)
ARB:TRAP:TOP? (@1:2)
-> 1.200000000E+01,5.000000000E+00
```

Related commands

`[SOURce:]ARB:TRAPezoid:STARt[:LEVel]`, `[SOURce:]ARB:TRAPezoid:TOP:TIME`, `[SOURce:]ARB:TRAPezoid:RTIMe`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.53 [SOURce:]ARB:TRIangle:FTIMe

Sets or queries the fall time of the triangle shape of a channel.

One execution is start level to top level and back: a linear rise over `ARB:TRIangle:RTIMe` followed by a linear fall over this time, which ends the execution at the start level. Rise and fall are programmed independently, so a sawtooth is a triangle with a very short fall time. The triangle has no hold phases at all; an application that needs dwell at a level uses `RAMP`, `TRAPezoid` or a user sequence. When playback starts the time is sampled onto the `ARB:TINTerval` grid; the programmed value is kept. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRIangle:FTIMe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	Duration of the transition from the top level back to the start level. 0 s omits the phase.
`MINimum`	CPD		Smallest programmable time (TBD).
`MAXimum`	CPD		Largest programmable time (TBD).
`DEFault`	CPD		Factory default, 1 ms.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRIangle:FTIMe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed fall time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRI:FTIM 5E-3
ARB:TRI:FTIM?
-> 5.000000000E-03
ARB:TRI:FTIM 1 ms,(@2)
ARB:TRI:FTIM? (@1:2)
-> 5.000000000E-03,1.000000000E-03
```

Related commands

[SOURce:]ARB:TRIangle:RTIMe, [SOURce:]ARB:TRIangle:TOP[:LEVel], [SOURce:]ARB:TINTerval, [SOURce:]ARB:SEQuence:COUNt

13.54 [SOURce:]ARB:TRIangle:RTIMe

Sets or queries the rise time of the triangle shape of a channel.

One execution starts at the start level, moves linearly to the top level over this time and returns over ARB:TRIangle:FTIMe; the length of one execution is the sum of the two times. The time is the duration of the transition itself and applies whether the top level lies above or below the start level. When playback starts the time is sampled onto the ARB:TINTerval grid; the programmed value is kept. A repeated triangle is periodic with the sum of the two times plus ARB:SEQuence:PAUSe; there is no frequency command. The setting is per channel.

COMMAND

```
[SOURce:]ARB:TRIangle:RTIMe
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<seconds>	NRf, s	TBD	Duration of the transition from the start level to the top level. 0 s omits the phase.
MINimum	CPD		Smallest programmable time (TBD).
MAXimum	CPD		Largest programmable time (TBD).
DEFault	CPD		Factory default, 1 ms.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRIangle:RTIMe?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable time instead of the setting.
MAXimum	CPD		Return the largest programmable time instead of the setting.
DEFault	CPD		Return the factory default, 1 ms, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, s`	The programmed rise time, not the value quantized onto the update grid, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The time is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The time is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The time is unchanged.

Example

```
ARB:TRI:RTIM 5E-3
ARB:TRI:RTIM?
-> 5.000000000E-03
ARB:TRI:RTIM 20 ms,(@2)
ARB:TRI:RTIM? (@1:2)
-> 5.000000000E-03,2.000000000E-02
```

Related commands

[SOURce:]ARB:TRIangle:FTIMe, [SOURce:]ARB:TRIangle:STARt[:LEVel], [SOURce:]ARB:SEQuence:COUNt, [SOURce:]ARB:TINTerval

13.55 [SOURce:]ARB:TRIangle:STARt[:LEVel]

Sets or queries the start level of the triangle shape of a channel.

The level is the voltage setpoint one execution starts at and the level the fall returns to, so it is both ends of the triangle. Native shapes modulate the voltage setpoint only; the current limit stays at its [SOURce:]CURRent value throughout playback. The triangle has no hold at this level: the rise over ARB:TRIangle:RTIMe begins immediately, and a repeated triangle passes through the level without dwelling on it. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:TRIangle:STARt[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	New voltage setpoint at the start and the end of the triangle.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRIangle:STARt[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the smallest programmable level instead of the setting.
`MAXimum`	CPD		Return the largest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed start level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation with a `TRANsient` participant is armed or active. The level is unchanged on every channel.
-222	`"Data out of range"`	`<value>` is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:TRI:STAR 0
ARB:TRI:STAR?
-> 0.000000000E+00
ARB:TRI:STAR 1,(@2)
ARB:TRI:STAR? (@1:2)
-> 0.000000000E+00,1.000000000E+00
```

Related commands

`[SOURce:]ARB:TRIangle:TOP[:LEVel]`, `[SOURce:]ARB:TRIangle:RTIMe`, `[SOURce:]ARB:TRIangle:FTIMe`, `[SOURce:]ARB:FUNCtion:SHAPe`

13.56 [SOURce:]ARB:TRIangle:TOP[:LEVel]

Sets or queries the top level of the triangle shape of a channel.

The level is the apex of the triangle: the setpoint reaches it at the end of `ARB:TRIangle:RTIMe` and immediately begins the fall of `ARB:TRIangle:FTIMe` back to the start level. Native shapes modulate the voltage setpoint only; the current limit stays at its `[SOURce:]CURRent` value throughout playback. A top level below the start level inverts the triangle; the apex is not held, so the waveform passes through the top level instead of dwelling on it. The setting is per channel. The query returns the programmed level, not a measurement.

COMMAND

```
[SOURce:]ARB:TRIangle:TOP[:LEVel]
  <value>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<value>`	NRf, V	Model-dependent	New voltage setpoint at the apex of the triangle.
`MINimum`	CPD		Smallest programmable level (model-dependent).
`MAXimum`	CPD		Largest programmable level (model-dependent).
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]ARB:TRIangle:TOP[:LEVel]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the smallest programmable level instead of the setting.
MAXimum	CPD		Return the largest programmable level instead of the setting.
DEFault	CPD		Return the factory default, 0 V, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, V — The programmed top level, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The level is unchanged on every channel.
-222	"Data out of range"	<value> is outside the programmable range, or a listed channel does not exist. The level is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The level is unchanged.

Example

```
ARB:TRI:TOP 5
ARB:TRI:TOP?
-> 5.000000000E+00
ARB:TRI:TOP 3.3,(@2)
ARB:TRI:TOP? (@1:2)
-> 5.000000000E+00,3.300000000E+00
```

Related commands

[SOURce:]ARB:TRIangle:STARt[:LEVel], [SOURce:]ARB:TRIangle:RTIMe, [SOURce:]ARB:TRIangle:FTIMe, [SOURce:]ARB:FUNCtion:SHAPe

13.57 [SOURce:]ARB[:STATe]

Selects or queries whether a channel takes part in the transient operation.

Participation has exactly one meaning: in ARB:MODE SEQuence only participating channels play their configured waveform, in STReam mode only participating channels occur in the stream points, in ascending channel order. It is not an output enable, not a trigger enable, not a playback start or stop and not a mode. Switching a channel off keeps its stored ARB configuration; writing the value a channel already has changes nothing, while an effective change of the participating set clears the DAC sample buffer because it changes the stream layout. INITiate with a TRANsient participant validates participating channels only and requires at least one, each with its output on. From a successful INITiate until the operation has ended, every participating channel is locked: its setpoints, sense source, output mode, protection settings and clears, measurement ranges and OUTPut ON generate -221, and only OUTPut OFF and ABORt remain. On the single-channel product the command carries no information, because the only channel is also the only possible participant.

COMMAND

```
[SOURce:]ARB[:STATe]
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		The channel takes part in the transient operation.
OFF | 0	Boolean		The channel is left out; its ARB configuration is kept.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
[SOURce:]ARB[:STATe]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when the channel participates, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with a TRANsient participant is armed or active. The participation and the DAC sample buffer are unchanged.
-222	"Data out of range"	A listed channel does not exist. The participation is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The participation is unchanged.

Example

```
ARB:STAT ON
ARB:STAT?
-> 1
ARB:STAT OFF,(@2)
ARB:STAT? (@1:2)
-> 1,0
```

Related commands

[SOURce:]ARB:MODE, INITiate, [SOURce:]ARB:STReam:DATA, [SOURce:]ARB:FUNCtion:SHAPe

13.58 [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

Sets or queries the current limit of a channel.

The limit is a non-negative magnitude. When the load current reaches it, the regulator switches from the voltage loop to the current loop and the channel reports CC in the Operation status; the output stays enabled, and reaching the limit is not a protection trip. The same magnitude limits current delivered to the DUT and current absorbed from the DUT in OUTPut:MODE AUTO or SINK. The setting is independent of the selected current measurement range and is accepted with the output on or off, but on a channel with ARB[:STATe] ON the limit is locked while an operation with a TRANsient participant is armed or active and the command generates -221; channels that do not take part in the playback, and every channel during an acquisition-only operation, accept it. The query returns the programmed limit, not a measurement; the load current is read with MEASure:CURRent?.

COMMAND

```
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]
  <current>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<current>`	NRf, A	Model-dependent	New current limit.
`MINimum`	CPD		Lowest programmable current limit of the channel.
`MAXimum`	CPD		Highest programmable current limit of the channel.
`DEFault`	CPD		Factory default, 0 A.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable limit instead of the setting.
`MAXimum`	CPD		Return the highest programmable limit instead of the setting.
`DEFault`	CPD		Return the factory default, 0 A, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, A` The programmed current limit, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). The limit is unchanged on every channel.
-222	`"Data out of range"`	`<current>` is outside the programmable range, or a listed channel does not exist. The limit is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The limit is unchanged.

Example

```
CURR 2.5
CURR?
-> 2.500000000E+00
CURR 500 mA,(@2)
CURR? (@1:2)
-> 2.500000000E+00,5.000000000E-01
```

Related commands

`[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]`, `MEASure[:SCALar]:CURRent[:DC]?`, `[SOURce:]CURRent:PROTection:LEVel`, `STATus:OPERation:CONDition?`

13.59 [SOURce:]CURRent:PROTection:CLEar

Clears the latched OCP trip of a channel.

The OCP enable state, level and delay are unchanged, and the output is not re-enabled; it stays off until it is enabled again with `OUTPut ON`. A latch that is not set is unchanged and causes no error. The OCP event already latched in the Questionable status is not erased; it is read and cleared with `STATus:QUEStionable[:EVENt]?` or *CLS. A trip whose electrical cause is still present prevents enabling the output after the clear. On a channel with `ARB[:STATe] ON` the clear is locked while an operation with a `TRANsient` participant is armed or active and generates -221.

COMMAND

```
[SOURce:]CURRent:PROTection:CLEar
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to clear. Omitted: channel 1. Every listed channel is cleared.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). No latch is cleared.
-222	"Data out of range"	A listed channel does not exist. No latch is cleared.
-171	"Invalid expression"	A channel appears twice in the channel list. No latch is cleared.

Example

```
CURR:PROT:TRIP?
-> 1
CURR:PROT:CLE
CURR:PROT:TRIP?
-> 0
OUTP ON
```

Related commands

`[SOURce:]CURRent:PROTection:TRIPped?`, `OUTPut:PROTection:CLEar`, `OUTPut[:STATe]`, `STATus:QUEStionable[:EVENt]?`

13.60 [SOURce:]CURRent:PROTection:DELay

Sets or queries the persistence delay of the overcurrent protection of a channel.

The load current must stay above the OCP level continuously for at least the delay before the trip is generated. When the excursion ends earlier, the timer restarts and no trip results. The delay covers the persistence of the condition only; the detection and turn-off latency after the protection decision is implementation-dependent and no maximum reaction time is specified. The setting is accepted with the output on or off and does not clear an already latched trip; on a channel with `ARB[:STATe] ON` it is locked while an operation with a `TRANsient` participant is armed or active and generates -221. The query returns the programmed delay.

COMMAND

```
[SOURce:]CURRent:PROTection:DELay
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	New persistence delay.
`MINimum`	CPD		Shortest programmable delay of the channel (TBD).
`MAXimum`	CPD		Longest programmable delay of the channel (TBD).
`DEFault`	CPD		Factory default, 0.05 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]CURRent:PROTection:DELay?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the shortest programmable delay instead of the setting.
MAXimum	CPD		Return the longest programmable delay instead of the setting.
DEFault	CPD		Return the factory default, 0.05 s, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s — The programmed persistence delay, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The delay is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The delay is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The delay is unchanged.

Example

```
CURR:PROT:DEL 0.1
CURR:PROT:DEL?
-> 1.000000000E-01
CURR:PROT:DEL? DEF
-> 5.000000000E-02
CURR:PROT:DEL 20 ms,(@2)
CURR:PROT:DEL? (@1:2)
-> 1.000000000E-01,2.000000000E-02
```

Related commands

[SOURce:]CURRent:PROTection:LEVel, [SOURce:]CURRent:PROTection[:STATe], [SOURce:]POWer:PROTection:DELay

13.61 [SOURce:]CURRent:PROTection:LEVel

Sets or queries the overcurrent protection threshold of a channel.

The threshold is independent of the current limit. Reaching the current limit switches the regulator to CC with the output enabled; a load current above the OCP level for longer than the OCP delay switches the output off and latches the trip. The level takes effect only while OCP is enabled, is accepted with the output on or off, and changing it does not clear an already latched trip; on a channel with `ARB[:STATe] ON` it is locked while an operation with a `TRANsient` participant is armed or active and generates -221. The query returns the programmed threshold, not a measurement.

COMMAND

```
[SOURce:]CURRent:PROTection:LEVel
  <current>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<current>`	NRf, A	Model-dependent	New OCP threshold.
`MINimum`	CPD		Lowest programmable OCP threshold of the channel.
`MAXimum`	CPD		Highest programmable OCP threshold of the channel.
`DEFault`	CPD		Factory default: the `MAXimum` of the model.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]CURRent:PROTection:LEVel?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable threshold instead of the setting.
`MAXimum`	CPD		Return the highest programmable threshold instead of the setting.
`DEFault`	CPD		Return the factory default, the `MAXimum` of the model, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, A`	The programmed OCP threshold, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). The threshold is unchanged on every channel.
-222	"Data out of range"	`<current>` is outside the programmable range, or a listed channel does not exist. The threshold is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The threshold is unchanged.

Example

```
CURR:PROT:LEV 3
CURR:PROT:LEV?
-> 3.000000000E+00
CURR:PROT:LEV 2.5,(@2)
CURR:PROT:LEV? (@1:2)
-> 3.000000000E+00,2.500000000E+00
```

Related commands

`[SOURce:]CURRent:PROTection[:STATe]`, `[SOURce:]CURRent:PROTection:DELay`, `[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]`, `[SOURce:]CURRent:PROTection:TRIPped?`

13.62 [SOURce:]CURRent:PROTection:TRIPped?

Returns whether the overcurrent protection of a channel has tripped and is still latched.

The latch is set when OCP switches the output off and stays set after the electrical condition has disappeared, until it is cleared with `CURRent:PROTection:CLEar`, `OUTPut:PROTection:CLEar` or *RST. Reading the latch does not change it. A tripped channel reports `0` in `OUTPut?`. The same trip sets the OCP bit of the Questionable status, so a controller can wait for a service request instead of polling this query.

QUERY

```
[SOURce:]CURRent:PROTection:TRIPped?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when the OCP trip is latched, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
CURR:PROT:TRIP?
-> 1
OUTP?
-> 0
CURR:PROT:TRIP? (@1:2)
-> 1,0
```

Related commands

[SOURce:]CURRent:PROTection:CLEar, STATus:QUEStionable:CONDition?, OUTPut[:STATe]

13.63 [SOURce:]CURRent:PROTection[:STATe]

Enables or disables the overcurrent protection of a channel, or queries its state.

With OCP enabled, a load current above the OCP level for longer than the OCP delay switches the output off and latches the trip. OCP is independent of the current limit, which stays active in either state. Disabling OCP stops new trips but does not clear an existing latch; the level and the delay are kept. The setting is accepted with the output on or off; on a channel with ARB[:STATe] ON it is locked while an operation with a TRANsient participant is armed or active and generates -221. OCP keeps working during DAC playback, and a trip ends the playback abnormally. Whether OCP evaluates externally imposed terminal conditions while the output is off is not yet specified (TBD).

COMMAND

```
[SOURce:]CURRent:PROTection[:STATe]
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ON | 1`	Boolean		Enable OCP.
`OFF | 0`	Boolean		Disable OCP.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
[SOURce:]CURRent:PROTection[:STATe]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR1>` 1 when OCP is enabled, otherwise `0`.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). The state is unchanged on every channel.
-222	`"Data out of range"`	A listed channel does not exist. The state is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The state is unchanged.

Example

```
CURR:PROT ON
CURR:PROT?
-> 1
CURR:PROT OFF,(@2)
CURR:PROT? (@1:2)
-> 1,0
```

Related commands

`[SOURce:]CURRent:PROTection:LEVel`, `[SOURce:]CURRent:PROTection:DELay`, `[SOURce:]CURRent:PROTection:TRIPped?`, `[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]`

13.64 [SOURce:]POWer:PROTection:CLEar

Clears the latched OPP trip of a channel.

The OPP enable state, level and delay are unchanged, and the output is not re-enabled; it stays off until it is enabled again with OUTPut ON. A latch that is not set is unchanged and causes no error. The OPP event already latched in the Questionable status is not erased; it is read and cleared with STATus:QUEStionable[:EVENt]? or *CLS. A trip whose electrical cause is still present prevents enabling the output after the clear. On a channel with ARB[:STATe] ON the clear is locked while an operation with a TRANsient participant is armed or active and generates -221.

COMMAND

```
[SOURce:]POWer:PROTection:CLEar
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to clear. Omitted: channel 1. Every listed channel is cleared.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). No latch is cleared.
-222	"Data out of range"	A listed channel does not exist. No latch is cleared.
-171	"Invalid expression"	A channel appears twice in the channel list. No latch is cleared.

Example

```
POW:PROT:TRIP?
-> 1
POW:PROT:CLE
POW:PROT:TRIP?
-> 0
OUTP ON
```

Related commands

[SOURce:]POWer:PROTection:TRIPped?, OUTPut:PROTection:CLEar, OUTPut[:STATe], STATus:QUEStionable[:EVENt]?

13.65 [SOURce:]POWer:PROTection:DELay

Sets or queries the persistence delay of the overpower protection of a channel.

The output power must stay above the OPP level continuously for at least the delay before the trip is generated. When the excursion ends earlier, the timer restarts and no trip results. The delay covers the persistence of the condition only; the detection and turn-off latency after the protection decision is implementation-dependent and no maximum reaction time is specified. The setting is accepted with the output on or off and does not clear an already latched trip; on a channel with `ARB[:STATe] ON` it is locked while an operation with a `TRANsient` participant is armed or active and generates `-221`. The query returns the programmed delay.

COMMAND

```
[SOURce:]POWer:PROTection:DELay
  <seconds>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<seconds>`	NRf, s	TBD	New persistence delay.
`MINimum`	CPD		Shortest programmable delay of the channel (TBD).
`MAXimum`	CPD		Longest programmable delay of the channel (TBD).
`DEFault`	CPD		Factory default, 0.05 s.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]POWer:PROTection:DELay?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the shortest programmable delay instead of the setting.
`MAXimum`	CPD		Return the longest programmable delay instead of the setting.
`DEFault`	CPD		Return the factory default, 0.05 s, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, s	The programmed persistence delay, or the selected capability value.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The delay is unchanged on every channel.
-222	"Data out of range"	<seconds> is outside the programmable range, or a listed channel does not exist. The delay is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The delay is unchanged.

Example

```
POW:PROT:DEL 0.2
POW:PROT:DEL?
-> 2.000000000E-01
POW:PROT:DEL? DEF
-> 5.000000000E-02
POW:PROT:DEL 50 ms,(@2)
POW:PROT:DEL? (@1:2)
-> 2.000000000E-01,5.000000000E-02
```

Related commands

[SOURce:]POWer:PROTection:LEVel, [SOURce:]POWer:PROTection[:STATe], [SOURce:]CURRent:PROTection:DELay

13.66 [SOURce:]POWer:PROTection:LEVel

Sets or queries the overpower protection threshold of a channel.

The threshold is a power in watts and is independent of the voltage setpoint and the current limit. An output power above the OPP level for longer than the OPP delay switches the output off and latches the trip. The level takes effect only while OPP is enabled, is accepted with the output on or off, and changing it does not clear an already latched trip; on a channel with ARB[:STATe] ON it is locked while an operation with a TRANsient participant is armed or active and generates -221. The query returns the programmed threshold, not a measurement; the output power is read with MEASure:POWer?.

COMMAND

```
[SOURce:]POWer:PROTection:LEVel
  <power>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<power>	NRf, W	Model-dependent	New OPP threshold.
MINimum	CPD		Lowest programmable OPP threshold of the channel.
MAXimum	CPD		Highest programmable OPP threshold of the channel.
DEFault	CPD		Factory default: the MAXimum of the model.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]POWer:PROTection:LEVel?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
MINimum	CPD		Return the lowest programmable threshold instead of the setting.
MAXimum	CPD		Return the highest programmable threshold instead of the setting.
DEFault	CPD		Return the factory default, the MAXimum of the model, instead of the setting.
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR3>, W — The programmed OPP threshold, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The threshold is unchanged on every channel.
-222	"Data out of range"	<power> is outside the programmable range, or a listed channel does not exist. The threshold is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The threshold is unchanged.

Example

```
POW:PROT:LEV 20
POW:PROT:LEV?
-> 2.000000000E+01
POW:PROT:LEV 10,(@2)
POW:PROT:LEV? (@1:2)
-> 2.000000000E+01,1.000000000E+01
```

Related commands

`[SOURce:]POWer:PROTection[:STATe]`, `[SOURce:]POWer:PROTection:DELay`, `[SOURce:]POWer:PROTection:TRIPped?`, `MEASure[:SCALar]:POWer?`

13.67 [SOURce:]POWer:PROTection:TRIPped?

Returns whether the overpower protection of a channel has tripped and is still latched.

The latch is set when OPP switches the output off and stays set after the electrical condition has disappeared, until it is cleared with `POWer:PROTection:CLEar`, `OUTPut:PROTection:CLEar` or *RST. Reading the latch does not change it. A tripped channel reports `0` in `OUTPut?`. The same trip sets the OPP bit of the Questionable status, so a controller can wait for a service request instead of polling this query.

QUERY

```
[SOURce:]POWer:PROTection:TRIPped?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR1>` 1 when the OPP trip is latched, otherwise `0`.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	A listed channel does not exist. No response is returned.
-171	`"Invalid expression"`	A channel appears twice in the channel list. No response is returned.

Example

```
POW:PROT:TRIP?
-> 1
OUTP?
-> 0
POW:PROT:TRIP? (@1:2)
-> 1,0
```

Related commands

`[SOURce:]POWer:PROTection:CLEar`, `STATus:QUEStionable:CONDition?`, `OUTPut[:STATe]`

13.68 [SOURce:]POWer:PROTection[:STATe]

Enables or disables the overpower protection of a channel, or queries its state.

With OPP enabled, an output power above the OPP level for longer than the OPP delay switches the output off and latches the trip. Disabling OPP stops new trips but does not clear an existing latch; the level and the delay are kept. The setting is accepted with the output on or off; on a channel with `ARB[:STATe] ON` it is locked while an operation with a `TRANsient` participant is armed or active and generates −221. OPP keeps working during DAC playback, and a trip ends the playback abnormally. Whether OPP evaluates externally imposed terminal conditions while the output is off is not yet specified (TBD).

COMMAND

```
[SOURce:]POWer:PROTection[:STATe]
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`ON | 1`	Boolean		Enable OPP.
`OFF | 0`	Boolean		Disable OPP.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
[SOURce:]POWer:PROTection[:STATe]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when OPP is enabled, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The state is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The state is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The state is unchanged.

Example

```
POW:PROT ON
POW:PROT?
-> 1
POW:PROT OFF,(@2)
POW:PROT? (@1:2)
-> 1,0
```

Related commands

[SOURce:]POWer:PROTection:LEVel, [SOURce:]POWer:PROTection:DELay, [SOURce:]POWer:PROTection:TRIPped?

13.69 [SOURce:]VOLTage:PROTection:CLEar

Clears the latched OVP trip of a channel.

The OVP enable state and level are unchanged, and the output is not re-enabled; it stays off until it is enabled again with OUTPut ON. A latch that is not set is unchanged and causes no error. The OVP event already latched in the Questionable status is not erased; it is read and cleared with STATus:QUEStionable[:EVENt]? or *CLS. A trip whose electrical cause is still present prevents enabling the output after the clear. On a channel with ARB[:STATe] ON the clear is locked while an operation with a TRANsient participant is armed or active and generates -221.

COMMAND

```
[SOURce:]VOLTage:PROTection:CLEar
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to clear. Omitted: channel 1. Every listed channel is cleared.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). No latch is cleared.
-222	"Data out of range"	A listed channel does not exist. No latch is cleared.
-171	"Invalid expression"	A channel appears twice in the channel list. No latch is cleared.

Example

```
VOLT:PROT:TRIP?
-> 1
VOLT:PROT:CLE
VOLT:PROT:TRIP?
-> 0
OUTP ON
```

Related commands

[SOURce:]VOLTage:PROTection:TRIPped?, OUTPut:PROTection:CLEar, OUTPut[:STATe], STATus:QUEStionable[:EVENt]?

13.70 [SOURce:]VOLTage:PROTection:LEVel

Sets or queries the overvoltage protection threshold of a channel.

The threshold is independent of the voltage setpoint. A terminal voltage above the OVP level switches the output off and latches the trip; OVP has no programmable delay, and the detection and turn-off latency is implementation-dependent. The level takes effect only while OVP is enabled, is accepted with the output on or off, and changing it does not clear an already latched trip; on a channel with ARB[:STATe] ON it is locked while an operation with a TRANsient participant is armed or active and generates -221. The query returns the programmed threshold, not a measurement.

COMMAND

```
[SOURce:]VOLTage:PROTection:LEVel
  <voltage>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<voltage>`	NRf, V	Model-dependent	New OVP threshold.
`MINimum`	CPD		Lowest programmable OVP threshold of the channel.
`MAXimum`	CPD		Highest programmable OVP threshold of the channel.
`DEFault`	CPD		Factory default: the `MAXimum` of the model.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]VOLTage:PROTection:LEVel?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable threshold instead of the setting.
`MAXimum`	CPD		Return the highest programmable threshold instead of the setting.
`DEFault`	CPD		Return the factory default, the `MAXimum` of the model, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` — The programmed OVP threshold, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
`-221`	`"Settings conflict"`	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). The threshold is unchanged on every channel.

-222	"Data out of range"	<voltage> is outside the programmable range, or a listed channel does not exist. The threshold is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The threshold is unchanged.

Example

```
VOLT:PROT:LEV 6
VOLT:PROT:LEV?
-> 6.000000000E+00
VOLT:PROT:LEV 13.5,(@2)
VOLT:PROT:LEV? (@1:2)
-> 6.000000000E+00,1.350000000E+01
```

Related commands

[SOURce:]VOLTage:PROTection[:STATe],
[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude],
[SOURce:]VOLTage:PROTection:TRIPped?

13.71 [SOURce:]VOLTage:PROTection:TRIPped?

Returns whether the overvoltage protection of a channel has tripped and is still latched.

The latch is set when OVP switches the output off and stays set after the electrical condition has disappeared, until it is cleared with VOLTage:PROTection:CLEar, OUTPut:PROTection:CLEar or *RST. Reading the latch does not change it. A tripped channel reports 0 in OUTPut?. The same trip sets the OVP bit of the Questionable status, so a controller can wait for a service request instead of polling this query.

QUERY

```
[SOURce:]VOLTage:PROTection:TRIPped?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1>	1 when the OVP trip is latched, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	A listed channel does not exist. No response is returned.
-171	"Invalid expression"	A channel appears twice in the channel list. No response is returned.

Example

```
VOLT:PROT:TRIP?
-> 1
OUTP?
-> 0
VOLT:PROT:TRIP? (@1:2)
-> 1,0
```

Related commands

[SOURce:]VOLTage:PROTection:CLEar, STATus:QUEStionable:CONDition?, OUTPut[:STATe]

13.72 [SOURce:]VOLTage:PROTection[:STATe]

Enables or disables the overvoltage protection of a channel, or queries its state.

With OVP enabled, a terminal voltage above the OVP level switches the output off and latches the trip. Disabling OVP stops new trips but does not clear an existing latch; the level is kept. The setting is accepted with the output on or off; on a channel with ARB[:STATe] ON it is locked while an operation with a TRANsient participant is armed or active and generates -221. OVP keeps working during DAC playback, and a trip ends the playback abnormally. Whether OVP evaluates externally imposed terminal conditions while the output is off is not yet specified (TBD).

COMMAND

```
[SOURce:]VOLTage:PROTection[:STATe]
  ON|OFF|1|0[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		Enable OVP.
OFF | 0	Boolean		Disable OVP.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same state is applied to every listed channel.

QUERY

```
[SOURce:]VOLTage:PROTection[:STATe]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<NR1> 1 when OVP is enabled, otherwise 0.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The state is unchanged on every channel.
-222	"Data out of range"	A listed channel does not exist. The state is unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The state is unchanged.

Example

```
VOLT:PROT ON
VOLT:PROT?
-> 1
VOLT:PROT OFF,(@2)
VOLT:PROT? (@1:2)
-> 1,0
```

Related commands

[SOURce:]VOLTage:PROTection:LEVel, [SOURce:]VOLTage:PROTection:TRIPped?, [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]

13.73 [SOURce:]VOLTage:SENSe[:SOURce]

Sets or queries the feedback point of the voltage regulation of a channel.

INTernal selects local two-wire sensing at the power terminals, EXTernal selects four-wire remote sensing at the sense terminals. Changing the sense source while the output is on switches that output off and leaves it off, so that the feedback path never changes under load; the output stays off until it is enabled again with OUTPut ON. On a channel with ARB[:STATe] ON the selection is locked while an operation with a TRANsient participant is armed or active: the command generates -221 and leaves sense source and output unchanged, so a sense change never ends a playback. Selecting EXTernal does not enable the output. The query returns the configured selection and has no side effects. The lock applies to the command, also when it repeats the current value. The lock applies to the command, also when it repeats the current value.

COMMAND

```
[SOURce:]VOLTage:SENSe[:SOURce]
  INTernal|EXTernal[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
INTernal	CPD		Local two-wire sensing at the power terminals.
EXTernal	CPD		Remote four-wire sensing at the sense terminals.
<chanlist>	Channel list		Channels to program. Omitted: channel 1. The same selection is applied to every listed channel.

QUERY

```
[SOURce:]VOLTage:SENSe[:SOURce]?
  [(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<chanlist>	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

<CRD>	INT for local two-wire sensing, EXT for remote four-wire sensing.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A listed channel has ARB[:STATe] ON and an operation with a TRANsient participant is armed or active (channel lock). The selection and the output state are unchanged.
-222	"Data out of range"	A listed channel does not exist. The selection and the output state are unchanged.
-171	"Invalid expression"	A channel appears twice in the channel list. The selection and the output state are unchanged.

Example

```
VOLT:SENS EXT
VOLT:SENS?
-> EXT
OUTP?
-> 0
OUTP ON
```

Related commands

OUTPut[:STATe], [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]

13.74 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]

Sets or queries the voltage setpoint of a channel.

The setpoint is the non-negative voltage the voltage loop regulates to while the load current stays below the current limit. It is independent of the selected voltage measurement range and is accepted with the output on or off, but on a channel with ARB[:STATe] ON the setpoint is locked while an operation with a TRANsient participant is armed or active and the command generates -221; channels that do not take part in the playback, and every channel during an acquisition-only operation, accept it. The query returns the programmed value, not a measurement; the terminal voltage is read with MEASure:VOLTage?.

COMMAND

```
[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]
  <voltage>|MINimum|MAXimum|DEFault[,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<voltage>`	NRf, V	Model-dependent	New setpoint.
`MINimum`	CPD		Lowest programmable voltage of the channel.
`MAXimum`	CPD		Highest programmable voltage of the channel.
`DEFault`	CPD		Factory default, 0 V.
`<chanlist>`	Channel list		Channels to program. Omitted: channel 1. The same value is applied to every listed channel.

QUERY

```
[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]?
  [MINimum|MAXimum|DEFault][,(@<chanlist>)]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable voltage instead of the setpoint.
`MAXimum`	CPD		Return the highest programmable voltage instead of the setpoint.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.
`<chanlist>`	Channel list		Channels to read. Omitted: channel 1. One value per listed channel, in list order.

Returns

`<NR3>, V` The programmed setpoint, or the selected capability value. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	A listed channel has `ARB[:STATe] ON` and an operation with a `TRANsient` participant is armed or active (channel lock). The setpoint is unchanged on every channel.
-222	`"Data out of range"`	`<voltage>` is outside the programmable range, or a listed channel does not exist. The setpoint is unchanged.
-171	`"Invalid expression"`	A channel appears twice in the channel list. The setpoint is unchanged.

Example

```
VOLT 5
VOLT?
-> 5.000000000E+00
VOLT 12,(@2)
VOLT? (@1:2)
-> 5.000000000E+00,1.200000000E+01
```

Related commands

`[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]`, `MEASure[:SCALar]:VOLTage[:DC]?`, `OUTPut[:STATe]`, `SENSe:VOLTage:RANGe`

14

CHAPTER 14

STATus commands

This chapter describes the three 16-bit status groups OPERation, QUEStionable and MEASurement. All three use the same mechanism: a device-wide condition register, a positive and a negative transition filter that decide which condition changes latch an event, a latched event register and an enable mask that routes the group into its Status Byte summary bit. The bit assignments of the three groups are listed in chapter 1; the numeric responses of all register queries follow FORMat:SREGister. STATus:PRESet restores the masks of all three groups; the STATus:QUEue commands read, clear and filter the error queue of the session.

IN THIS CHAPTER

14.1 STATus:OPERation:CONDition? 285
14.2 STATus:OPERation:ENABle 285
14.3 STATus:OPERation:NTRansition 287
14.4 STATus:OPERation:PTRansition 288
14.5 STATus:OPERation[:EVENt]? 289
14.6 STATus:QUEStionable:CONDition? 290
14.7 STATus:QUEStionable:ENABle 291
14.8 STATus:QUEStionable:NTRansition 292
14.9 STATus:QUEStionable:PTRansition 293
14.10 STATus:QUEStionable[:EVENt]? 294
14.11 STATus:MEASurement:CONDition? 295
14.12 STATus:MEASurement:ENABle 295
14.13 STATus:MEASurement:NTRansition 297
14.14 STATus:MEASurement:PTRansition 298
14.15 STATus:MEASurement[:EVENt]? 299
14.16 STATus:PRESet 300
14.17 STATus:QUEue:CLEar 300
14.18 STATus:QUEue:DISable 301

14.19 STATus:QUEue:ENABle 302

14.20 STATus:QUEue[:NEXT]? 303

14.1 STATus:OPERation:CONDition?

Returns the live Operation condition register.

The Operation condition reports normal regulation and power-flow state, not faults: CV, CC, sourcing and sinking. The bits are listed in chapter 1. The register is device-wide and identical for every session; reading it is non-destructive and changes neither the condition, the latched event register nor any mask. With the output off all Operation condition bits are 0. Reserved bits are always 0.

QUERY

```
STATus:OPERation:CONDition?
```

Returns

<NR1>	The current Operation condition as a 16-bit value, in the format selected by FORMat:SREGister.

Example

```
STAT:OPER:COND?
-> 5
```

Related commands

STATus:OPERation[:EVENt]?, STATus:OPERation:ENABle, STATus:QUEStionable:CONDition?, FORMat:SREGister

14.2 STATus:OPERation:ENABle

Sets or queries which latched Operation events are summarized in the Status Byte.

A bit set in the mask lets the corresponding event bit contribute to the OPER summary bit 7 of the Status Byte, which *SRE can turn into a service request. The mask does not control whether events are latched; that is the task of PTRansition and NTRansition. Writing the mask never synthesizes an event from the current condition and never clears an already latched event. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:OPERation:ENABle
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Event bits that contribute to the OPER summary bit. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:OPERation:ENABle?
```

Returns

<NR1> The current enable mask, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The mask is unchanged.

Example

```
STAT:OPER:ENAB 2
STAT:OPER:ENAB?
-> 2
```

Related commands

STATus:OPERation[:EVENt]?, STATus:OPERation:PTRansition, STATus:PRESet, *SRE

14.3 STATus:OPERation:NTRansition

Sets or queries which falling Operation condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from 1 to 0. The negative filter is independent of PTRansition: both, one or neither may be set for a bit, and a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition; only a subsequent qualifying transition creates one. Latching is independent of ENABle, which only routes latched events into the Status Byte. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:OPERation:NTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Condition bits whose 1 to 0 transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:OPERation:NTRansition?
```

Returns

<NR1> The current negative transition filter, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The filter is unchanged.

Example

```
STAT:OPER:NTR 2
STAT:OPER:NTR?
-> 2
```

Related commands

`STATus:OPERation:PTRansition`, `STATus:OPERation[:EVENt]?`, `STATus:OPERation:ENABle`, `STATus:PRESet`

14.4 STATus:OPERation:PTRansition

Sets or queries which rising Operation condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from `0` to `1`. The positive filter is independent of `NTRansition`; a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition, and it does not clear events already latched. Latching is independent of `ENABle`, which only routes latched events into the Status Byte. The mask belongs to the calling session; `STATus:PRESet` restores the default.

COMMAND

```
STATus:OPERation:PTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<mask>`	NR1	0 to 65535	Condition bits whose `0` to `1` transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:OPERation:PTRansition?
```

Returns

`<NR1>` The current positive transition filter, in the format selected by `FORMat:SREGister`.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<mask>` is outside `0` to `65535`. The filter is unchanged.

Example

```
STAT:OPER:PTR 3
STAT:OPER:PTR?
-> 3
```

Related commands

STATus:OPERation:NTRansition, STATus:OPERation[:EVENt]?, STATus:OPERation:ENABle, STATus:PRESet

14.5 STATus:OPERation[:EVENt]?

Returns and clears the latched Operation event register.

An event bit is latched when the corresponding condition bit performs a transition selected by PTRansition or NTRansition, independently of the ENABle mask. The query returns the register and then clears exactly the bits it reported; the condition register is not affected, so a condition that is still present is reported again by CONDition?. The event register belongs to the calling session, so reading it never consumes the event history of another session. *CLS clears it as well; STATus:PRESet does not. The bits are listed in chapter 1.

QUERY

```
STATus:OPERation[:EVENt]?
```

Returns

<NR1>	The latched Operation events since the last read, in the format selected by FORMat:SREGister.

Example

```
STAT:OPER:EVEN?
-> 2
STAT:OPER:EVEN?
-> 0
```

Related commands

STATus:OPERation:CONDition?, STATus:OPERation:ENABle, STATus:OPERation:PTRansition, *CLS

14.6 STATus:QUEStionable:CONDition?

Returns the live Questionable condition register.

The Questionable condition reports latched protection and terminal fault state: the OVP, OCP and OPP trips, overtemperature, the inhibit input as bit 4 INHIBIT, a hardware fault as bit 5 HW_FAULT, acquisition overrun, DAC underrun and one common storage error. The bits are listed in chapter 1. The OVP, OCP and OPP bits clear through their own ...:PROTection:CLEar command or through OUTPut:PROTection:CLEar, which besides *RST is the only clear command for OTP, INHIBIT and HW_FAULT: OTP stays set after the instrument has cooled down and becomes clearable once every channel is back in its normal temperature range, HW_FAULT becomes clearable once the hardware has recovered by itself, and while the instrument is still hot or the fault persists both stay set and a clear has no effect. INHIBIT follows the inhibit input in LIVE mode, every activation being a new positive transition, and in LATChing mode is set whenever the input is active, also when it is already active as the line is routed or the mode or level is set, and stays set until OUTPut:PROTection:CLEar clears it with the input inactive. The register is device-wide: a bit is set while its condition is present on at least one channel and clears when it is clear on every channel; the affected channel is read with the ...:PROTection:TRIPped? queries. OTP, INHIBIT and HW_FAULT switch every output off. The overrun, underrun and storage bits stay asserted until an operation of the same kind is successfully initiated again. *RST clears these bits as well; OTP and HW_FAULT stay set while their cause persists, a clear has no effect then, and INHIBIT is cleared because *RST also selects OUTPut:INHibit:MODE OFF, after which the input no longer counts. *CLS never clears condition bits. The register is device-wide and reading it is non-destructive.

QUERY

```
STATus:QUEStionable:CONDition?
```

Returns

<NR1>	The current Questionable condition as a 16-bit value, in the format selected by FORMat:SREGister.

Example

```
STAT:QUES:COND?
-> 2
```

Related commands

STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:ENABle, OUTPut:PROTection:CLEar, SYSTem:ERRor[:NEXT]?

14.7 STATus:QUEStionable:ENABle

Sets or queries which latched Questionable events are summarized in the Status Byte.

A bit set in the mask lets the corresponding event bit contribute to the QUES summary bit 3 of the Status Byte, which *SRE can turn into a service request. The mask does not control whether events are latched; that is the task of PTRansition and NTRansition. Writing the mask never synthesizes an event from the current condition and never clears an already latched event. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:QUEStionable:ENABle
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Event bits that contribute to the QUES summary bit. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:QUEStionable:ENABle?
```

Returns

<NR1> The current enable mask, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The mask is unchanged.

Example

```
STAT:QUES:ENAB 511
STAT:QUES:ENAB?
-> 511
```

Related commands

STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:PTRansition, STATus:PRESet, *SRE

14.8 STATus:QUEStionable:NTRansition

Sets or queries which falling Questionable condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from 1 to 0, for example when a protection trip is cleared. The negative filter is independent of PTRansition: both, one or neither may be set for a bit, and a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition. Latching is independent of ENABle, which only routes latched events into the Status Byte. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:QUEStionable:NTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Condition bits whose 1 to 0 transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:QUEStionable:NTRansition?
```

Returns

<NR1> The current negative transition filter, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The filter is unchanged.

Example

```
STAT:QUES:NTR 0
STAT:QUES:NTR?
-> 0
```

Related commands

`STATus:QUEStionable:PTRansition`, `STATus:QUEStionable[:EVENt]?`, `STATus:QUEStionable:ENABle`, `STATus:PRESet`

14.9 STATus:QUEStionable:PTRansition

Sets or queries which rising Questionable condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from `0` to `1`, which is the usual way to catch a protection trip or a terminal fault. The positive filter is independent of `NTRansition`; a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition, so a trip that is already latched in the condition register is not reported retroactively. The mask belongs to the calling session; `STATus:PRESet` restores the default.

COMMAND

```
STATus:QUEStionable:PTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<mask>`	NR1	0 to 65535	Condition bits whose `0` to `1` transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:QUEStionable:PTRansition?
```

Returns

`<NR1>` The current positive transition filter, in the format selected by `FORMat:SREGister`.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	`<mask>` is outside `0` to `65535`. The filter is unchanged.

Example

```
STAT:QUES:PTR 511
STAT:QUES:PTR?
-> 511
```

Related commands

STATus:QUEStionable:NTRansition, STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:ENABle, STATus:PRESet

14.10 STATus:QUEStionable[:EVENt]?

Returns and clears the latched Questionable event register.

An event bit is latched when the corresponding condition bit performs a transition selected by PTRansition or NTRansition, independently of the ENABle mask. The query returns the register and then clears exactly the bits it reported; the condition register is not affected, so a protection trip that is still latched remains visible in CONDition? and still blocks the output. The event register belongs to the calling session and is not consumed by other sessions. *CLS clears it as well; STATus:PRESet does not, and an event produced by a protection trip stays latched after the trip has been cleared until it is read or cleared with *CLS. The bits are listed in chapter 1.

QUERY

```
STATus:QUEStionable[:EVENt]?
```

Returns

<NR1>	The latched Questionable events since the last read, in the format selected by FORMat:SREGister.

Example

```
STAT:QUES:EVEN?
-> 2
STAT:QUES:EVEN?
-> 0
```

Related commands

STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABle, STATus:QUEStionable:PTRansition, *CLS

14.11 STATus:MEASurement:CONDition?

Returns the live Measurement condition register.

The Measurement condition is the compact live state of the triggered operation: waiting for the trigger, active, one ADC block ready, room for one complete DAC stream block, and a generic reading overflow. The bits are listed in chapter 1. ADC_BUFFER_READY and DAC_BUFFER_FREE are guarantees, not hints: while ADC_BUFFER_READY is set, one FETCh:ARRay:NEXT? block is returned without waiting for buffer progress, and while DAC_BUFFER_FREE is set, one complete 4096-point stream block is accepted without waiting for playback to free buffer space. There is no DONE bit; normal completion is the falling edge of ACTIVE. The register is device-wide and reading it is non-destructive. ACTIVE stays 1 while a DLog file is still being flushed and closed after ABORt or after the acquisition has stopped, and clears when the file is closed.

QUERY

```
STATus:MEASurement:CONDition?
```

Returns

<NR1>	The current Measurement condition as a 16-bit value, in the format selected by FORMat:SREGister.

Example

```
STAT:MEAS:COND?
-> 6
```

Related commands

STATus:MEASurement[:EVENt]?, STATus:MEASurement:NTRansition, FETCh:ARRay:NEXT?, INITiate

14.12 STATus:MEASurement:ENABle

Sets or queries which latched Measurement events are summarized in the Status Byte.

A bit set in the mask lets the corresponding event bit contribute to the MEAS summary bit 0 of the Status Byte, which *SRE can turn into a service request. This is the usual completion path: enable bit 1 here and in NTRansition to be notified when ACTIVE falls. The mask does not control whether events are latched, and writing it neither synthesizes an event from the current condition nor clears a latched one. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:MEASurement:ENABle
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Event bits that contribute to the MEAS summary bit. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:MEASurement:ENABle?
```

Returns

<NR1> The current enable mask, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The mask is unchanged.

Example

```
STAT:MEAS:ENAB 2
STAT:MEAS:ENAB?
-> 2
```

Related commands

STATus:MEASurement[:EVENt]?, STATus:MEASurement:NTRansition, STATus:PRESet, *SRE

14.13 STATus:MEASurement:NTRansition

Sets or queries which falling Measurement condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from 1 to 0. Bit 1 is the important one: the falling edge of ACTIVE is how the completion of a triggered operation is signalled, since no dedicated completion bit exists. The negative filter is independent of PTRansition, and a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:MEASurement:NTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Condition bits whose 1 to 0 transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:MEASurement:NTRansition?
```

Returns

<NR1> The current negative transition filter, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The filter is unchanged.

Example

```
STAT:MEAS:NTR 2
STAT:MEAS:NTR?
-> 2
```

Related commands

STATus:MEASurement:ENABle, STATus:MEASurement[:EVENt]?, STATus:MEASurement:PTRansition, *OPC

14.14 STATus:MEASurement:PTRansition

Sets or queries which rising Measurement condition transitions latch an event.

A bit set in the mask latches the corresponding event bit when that condition bit changes from 0 to 1, for example when the operation is armed, when a block becomes fetchable or when a reading overflow occurs. The positive filter is independent of NTRansition; a bit set in neither filter never produces an event. Writing the mask does not synthesize an event from the current condition. Latching is independent of ENABle, which only routes latched events into the Status Byte. The mask belongs to the calling session; STATus:PRESet restores the default.

COMMAND

```
STATus:MEASurement:PTRansition
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<mask>	NR1	0 to 65535	Condition bits whose 0 to 1 transition latches an event. Bits of reserved condition bits are accepted and have no effect.

QUERY

```
STATus:MEASurement:PTRansition?
```

Returns

<NR1> The current positive transition filter, in the format selected by FORMat:SREGister.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<mask> is outside 0 to 65535. The filter is unchanged.

Example

```
STAT:MEAS:PTR 16
STAT:MEAS:PTR?
-> 16
```

Related commands

STATus:MEASurement:NTRansition, STATus:MEASurement[:EVENt]?, STATus:MEASurement:ENABle, STATus:PRESet

14.15 STATus:MEASurement[:EVENt]?

Returns and clears the latched Measurement event register.

An event bit is latched when the corresponding condition bit performs a transition selected by PTRansition or NTRansition, independently of the ENABle mask. The query returns the register and then clears exactly the bits it reported; the condition register is not affected, so a still-running operation keeps ACTIVE set in CONDition?. A latched bit 1 read after the operation has finished is the completion notification; the live condition bit is already 0 at that point. The event register belongs to the calling session. *CLS clears it as well; STATus:PRESet does not. The bits are listed in chapter 1.

QUERY

```
STATus:MEASurement[:EVENt]?
```

Returns

<NR1>	The latched Measurement events since the last read, in the format selected by FORMat:SREGister.

Example

```
STAT:MEAS:EVEN?
-> 2
STAT:MEAS:EVEN?
-> 0
```

Related commands

STATus:MEASurement:CONDition?, STATus:MEASurement:NTRansition, STATus:MEASurement:ENABle, *CLS

14.16 STATus:PRESet

Restores the enable masks and transition filters of the three status groups to their defaults.

For all three groups alike the command sets ENABle to 0, PTRansition to 65535 and NTRansition to 0. Only the masks of the calling session are affected. The command clears no event or condition register; the Standard Event register, *ESE, *SRE, the error queue with its STATus:QUEue:ENABle filter and the instrument configuration are untouched, and no event is generated. *CLS is the complementary command: it clears event history and the error queue but leaves the masks unchanged.

COMMAND

```
STATus:PRESet
```

Example

```
STAT:PRES
STAT:QUES:ENAB?
-> 0
STAT:QUES:PTR?
-> 65535
STAT:QUES:NTR?
-> 0
```

Related commands

STATus:QUEStionable:ENABle, STATus:OPERation:PTRansition, *CLS, *SRE

14.17 STATus:QUEue:CLEar

Clears the error queue of the calling session.

The command is an alias of SYSTem:ERRor:CLEar and acts on the same per-session error queue; no second error store exists. All pending entries are discarded, SYSTem:ERRor:COUNt? then returns 0 and the EAV bit 2 of the Status Byte clears. The error queues of other sessions are not touched. Unlike *CLS the command affects only the error queue: event registers, the Standard Event register, condition registers and all masks are unchanged.

COMMAND

```
STATus:QUEue:CLEar
```

Example

```
STAT:QUE:CLE
SYST:ERR:COUN?
-> 0
```

Related commands

SYSTem:ERRor:CLEar, SYSTem:ERRor[:NEXT]?, SYSTem:ERRor:COUNt?, *CLS

14.18 STATus:QUEue:DISable

Selects or queries the error codes that are discarded instead of queued for the calling session.

The error filter of the calling session is replaced so that every error code is queued except the listed ones; the command is the complement of STATus:QUEue:ENABle. A discarded error still prevents execution of the offending command and still sets its Standard Event bit; only the queue entry and the EAV bit are suppressed. Codes the instrument does not know are accepted and have no effect. Without a list every code is queued again. The filter is volatile and belongs to the calling session; *CLS and STATus:PRESet do not change it.

COMMAND

```
STATus:QUEue:DISable
  [<code>[,<code>...]]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<code>	NR1		Error code that is discarded; several codes are separated by commas. Omitted: every code is queued.

QUERY

```
STATus:QUEue:DISable?
```

Returns

<NR1>[,<NR1>...] The comma-separated codes currently discarded for the calling session, or 0 when every code is queued.

Example

```
STAT:QUE:DIS -222
VOLT 1E9
SYST:ERR?
-> 0,"No error"
STAT:QUE:DIS?
-> -222
STAT:QUE:DIS
STAT:QUE:DIS?
-> 0
```

Related commands

STATus:QUEue:ENABle, SYSTem:ERRor[:NEXT]?, STATus:QUEue[:NEXT]?, *ESR?

14.19 STATus:QUEue:ENABle

Selects or queries the error codes that are queued for the calling session.

The error filter of the calling session is replaced so that exactly the listed error codes are queued; every other error is discarded before it reaches the queue. A discarded error still prevents execution of the offending command and still sets its Standard Event bit; only the queue entry and the EAV bit are suppressed. Codes the instrument does not know are accepted and have no effect. Without a list no code is queued. The filter is volatile and belongs to the calling session; *CLS and STATus:PRESet do not change it.

COMMAND

```
STATus:QUEue:ENABle
  [<code>[,<code>...]]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<code>	NR1		Error code that is queued; several codes are separated by commas. Omitted: no code is queued.

QUERY

```
STATus:QUEue:ENABle?
```

Returns

`<NR1>[,<NR1>...]`	The comma-separated codes currently queued for the calling session, or `0` when no code is queued.

Example

```
STAT:QUE:ENAB -222,-224
STAT:QUE:ENAB?
-> -222,-224
STAT:QUE:ENAB
STAT:QUE:ENAB?
-> 0
```

Related commands

`STATus:QUEue:DISable`, `SYSTem:ERRor[:NEXT]?`, `STATus:QUEue:CLEar`, `*ESR?`

14.20 STATus:QUEue[:NEXT]?

Returns and removes the oldest entry of the error queue of the calling session.

The query is an alias of `SYSTem:ERRor[:NEXT]?`: it reads the same per-session error queue and returns the same response. Entries are returned oldest first and are removed when read; an empty queue returns `0,"No error"`. Reading the last entry clears the EAV bit 2 of the Status Byte. Errors caused by a program message are queued in the session that sent it, and asynchronous errors of an operation in the session that started it, so the queue read here is not a device-wide error log. The response is text and is not affected by `FORMat:SREGister`.

QUERY

```
STATus:QUEue[:NEXT]?
```

Returns

`<NR1>,<String>`	The error code and the message in quotes of the oldest entry, or `0,"No error"` when the queue is empty.

Example

```
STAT:QUE?
-> -222,"Data out of range"
STAT:QUE?
-> 0,"No error"
```

Related commands

`SYSTem:ERRor[:NEXT]?`, `SYSTem:ERRor:COUNt?`, `STATus:QUEue:CLEar`, `*CLS`

15

CHAPTER 15

SYSTem commands

This chapter describes the instrument and communication settings: the LAN configuration with its pending and active values, IPv4 and IPv6 addressing, the USB device class, the RS-232 interface and the LXI event domain, the real-time clock and time zone, the mains frequency, the error queue of each session, local and remote control, CSV file formatting, the beeper and the SCPI version. Most of these settings are persistent: they survive *RST, *SAV and *RCL and are changed only by their own commands.

IN THIS CHAPTER

15.1 SYSTem:BEEPer:STATe 307

15.2 SYSTem:BEEPer[:IMMediate] 308

15.3 SYSTem:COMMunicate:LAN:APPLy 309

15.4 SYSTem:COMMunicate:LAN:DESCription 309

15.5 SYSTem:COMMunicate:LAN:DESCription:STATus? 311

15.6 SYSTem:COMMunicate:LAN:DISCard 311

15.7 SYSTem:COMMunicate:LAN:HOSTname 312

15.8 SYSTem:COMMunicate:LAN:HOSTname:STATus? 313

15.9 SYSTem:COMMunicate:LAN:IPV4:ADDRess 314

15.10 SYSTem:COMMunicate:LAN:IPV4:GATeway 315

15.11 SYSTem:COMMunicate:LAN:IPV4:MODE 316

15.12 SYSTem:COMMunicate:LAN:IPV4:STATus? 317

15.13 SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess? 318

15.14 SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway? 319

15.15 SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask? 319

15.16 SYSTem:COMMunicate:LAN:IPV4:SUBNetmask 320

15.17 SYSTem:COMMunicate:LAN:IPV6:ADDRess 321

15.18 SYSTem:COMMunicate:LAN:IPV6:GATeway 322

15.19 SYSTem:COMMunicate:LAN:IPV6:MODE 323

15.20 SYSTem:COMMunicate:LAN:IPV6:PREFix 324

15.21 SYSTem:COMMunicate:LAN:IPV6:STATus? 325

15.22 SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal? 326

15.23 SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal? 327

15.24 SYSTem:COMMunicate:LAN:MACaddress? 328

15.25 SYSTem:COMMunicate:LAN:RESet 328

15.26 SYSTem:COMMunicate:LXI:EVENt:DOMain 329

15.27 SYSTem:COMMunicate:RLSTate 330

15.28 SYSTem:COMMunicate:SELF? 331

15.29 SYSTem:COMMunicate:SERial:BAUDrate 332

15.30 SYSTem:COMMunicate:SERial:FLOWcontrol 333

15.31 SYSTem:COMMunicate:USB:CLASs 334

15.32 SYSTem:CSV:DECimal 335

15.33 SYSTem:CSV:DELimiter 336

15.34 SYSTem:CSV:LINeending 337

15.35 SYSTem:DATE 338

15.36 SYSTem:ERRor:ALL? 339

15.37 SYSTem:ERRor:CLEar 340

15.38 SYSTem:ERRor:CODE:ALL? 341

15.39 SYSTem:ERRor:CODE[:NEXT]? 341

15.40 SYSTem:ERRor:COUNt? 342

15.41 SYSTem:ERRor[:NEXT]? 343

15.42 SYSTem:LFRequency 344

15.43 SYSTem:LOCal 345

15.44 SYSTem:LOCK:OWNer? 346

15.45 SYSTem:REMote 346

15.46 SYSTem:RWLock 347

15.47 SYSTem:TIME 348

15.48 SYSTem:TIME:PTP:STATus? 349

15.49 SYSTem:TIME:SOURce? 350

15.50 SYSTem:TZONe 350

15.51 SYSTem:VERSion? 352

15.1 SYSTem:BEEPer:STATe

Enables or disables the automatically generated audible feedback of the instrument.

The state covers only sounds the instrument produces on its own, such as key and touch feedback and warning or error tones. Tones requested explicitly with SYSTem:BEEPer[:IMMediate] are not suppressed by OFF and are produced regardless of this setting. Audible feedback is never the only indication of a protection or fault condition, so switching it off loses no safety information; the Questionable status continues to report every protection and fault condition.

COMMAND

```
SYSTem:BEEPer:STATe
  ON|OFF|1|0
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
ON | 1	Boolean		Automatic audible feedback is produced. Factory default.
OFF | 0	Boolean		Automatic audible feedback is suppressed. Explicitly requested tones are still produced.

QUERY

```
SYSTem:BEEPer:STATe?
```

Returns

<NR1> 1 when automatic audible feedback is enabled, otherwise 0.

Example

```
SYST:BEEP:STAT OFF
SYST:BEEP:STAT?
-> 0
```

Related commands

SYSTem:BEEPer[:IMMediate], DISPlay:BRIGhtness

15.2 SYSTem:BEEPer[:IMMediate]

Generates one tone on the instrument beeper immediately.

Without parameters the instrument produces one product-defined standard tone; with the two parameters it produces a tone of <frequency> hertz for <duration> seconds. Both forms are executed even while BEEPer:STATe is OFF, because the tone was requested explicitly by the controller, and neither form changes BEEPer:STATe, which governs only the automatic sounds of the instrument. The command changes no source, measurement, acquisition, trigger or status configuration and has no query form. The supported frequency and duration limits are hardware constants of the model that are not yet fixed (TBD).

COMMAND

```
SYSTem:BEEPer[:IMMediate]
  [<frequency>,<duration>]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<frequency>	NRf, Hz	TBD	Tone frequency. Omitted together with <duration>: the product-defined standard tone is produced.
<duration>	NRf, s	TBD	Length of the tone. Given only together with <frequency>.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<frequency> or <duration> is outside the supported range of the model. No tone is produced.

Example

```
SYST:BEEP
SYST:BEEP:STAT OFF
SYST:BEEP 1000,0.2
```

Related commands

SYSTem:BEEPer:STATe

15.3 SYSTem:COMMunicate:LAN:APPLy

Applies the staged LAN configuration to the Ethernet interface.

Every LAN setter writes the staged configuration only, so a complete static configuration can be prepared before the running network configuration is replaced; APPLy commits all staged LAN settings together, reconfigures the Ethernet interface and restarts the affected LAN services. The applied configuration becomes the new persistent LAN configuration. When the new configuration changes an address, a protocol mode, the hostname or another property the current remote connection depends on, that connection may be lost; this is a normal consequence of applying the configuration, not a fault. USB, RS-232 and LXI event settings are not staged and are not affected.

COMMAND

```
SYSTem:COMMunicate:LAN:APPLy
```

Example

```
SYST:COMM:LAN:IPV4:MODE MAN
SYST:COMM:LAN:IPV4:ADDR ”192.168.1.50”
SYST:COMM:LAN:IPV4:SUBN ”255.255.255.0”
SYST:COMM:LAN:IPV4:GAT ”192.168.1.1”
SYST:COMM:LAN:APPL
SYST:COMM:LAN:IPV4:STAT?
-> STATIC
```

Related commands

SYSTem:COMMunicate:LAN:DISCard, SYSTem:COMMunicate:LAN:RESet, SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:IPV4:STATus?

15.4 SYSTem:COMMunicate:LAN:DESCription

Sets or queries the DNS-SD service-instance base name of the instrument.

The value is the human-readable base name under which the instrument advertises its LAN services through mDNS/DNS-SD; it is a discovery name, not a free-form instrument note and not the model description of *IDN?. Up to 63 printable ASCII characters are accepted. The setter writes the staged LAN configuration and becomes active with LAN:APPLy. The query returns the staged base name, which is not necessarily the name advertised on the network: mDNS conflict resolution may append a numeric suffix, and the resulting name is read with LAN:DESCription:STATus?.

COMMAND

```
SYSTem:COMMunicate:LAN:DESCription
  <name>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<name>	String		Service-instance base name in quotes, up to 63 printable ASCII characters.

QUERY

```
SYSTem:COMMunicate:LAN:DESCription?
```

Returns

<String> The staged service-instance base name, in quotes.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	<name> is longer than 63 characters or contains a non-printable character. The staged base name is unchanged.

Example

```
SYST:COMM:LAN:DESC "LPS3010 Bench 4"
SYST:COMM:LAN:APPL
SYST:COMM:LAN:DESC?
-> "LPS3010 Bench 4"
```

Related commands

SYSTem:COMMunicate:LAN:DESCription:STATus?, SYSTem:COMMunicate:LAN:HOSTname, SYSTem:COMMunicate:LAN:APPLy

15.5 SYSTem:COMMunicate:LAN:DESCription:STATus?

Returns the DNS-SD service-instance name currently advertised on the network.

The advertised name is derived from the applied service description; when another device on the link already uses that name, mDNS conflict resolution appends a numerical suffix, and the resolved name is what this query returns. Without a conflict the response is identical to LAN:DESCription?. The query reports the state of the running interface and is therefore independent of staged values that have not been applied. Reading it changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:DESCription:STATus?
```

Returns

<String>	The service-instance name currently advertised, in quotes.

Example

```
SYST:COMM:LAN:DESC?
-> "LPS3010 Bench 4"
SYST:COMM:LAN:DESC:STAT?
-> "LPS3010 Bench 4 (2)"
```

Related commands

SYSTem:COMMunicate:LAN:DESCription, SYSTem:COMMunicate:LAN:HOSTname:STATus?, SYSTem:COMMunicate:LAN:APPLy

15.6 SYSTem:COMMunicate:LAN:DISCard

Discards the staged LAN changes and restores the staged values from the applied configuration.

All LAN changes made since the last successful LAN:APPLy or LAN:RESet are dropped, so a subsequent LAN configuration query returns the applied value again. The Ethernet interface is not reconfigured, the LAN services are not restarted and an active remote connection is not disturbed. Settings that are never staged, such as the USB class, the RS-232 parameters and the LXI event domain, are not affected. Discarding when nothing has been staged is harmless.

COMMAND

```
SYSTem:COMMunicate:LAN:DISCard
```

Example

```
SYST:COMM:LAN:IPV4:MODE MAN
SYST:COMM:LAN:IPV4:MODE?
-> MAN
SYST:COMM:LAN:DISC
SYST:COMM:LAN:IPV4:MODE?
-> AUTO
```

Related commands

SYSTem:COMMunicate:LAN:APPLy, SYSTem:COMMunicate:LAN:RESet, SYSTem:COMMunicate:LAN:IPV4:MODE

15.7 SYSTem:COMMunicate:LAN:HOSTname

Sets or queries the base hostname of the Ethernet interface.

The hostname is used by the Ethernet interface and by mDNS as the base host label; an empty string selects the product-defined default hostname. The setter writes the staged LAN configuration and becomes active with LAN:APPLy. The query returns the staged base hostname without a conflict suffix that mDNS may have added, which is read with LAN:HOSTname:STATus?.

COMMAND

```
SYSTem:COMMunicate:LAN:HOSTname
  <hostname>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<hostname>	String		Base host label in quotes, at most 15 characters from the set of ASCII letters, digits and -, beginning and ending alphanumeric. An empty string selects the product-defined default hostname.

QUERY

```
SYSTem:COMMunicate:LAN:HOSTname?
```

Returns

<String>	The staged base hostname, in quotes.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	<hostname> violates the host-label rules. The staged hostname is unchanged.

Example

```
SYST:COMM:LAN:HOST "lps-bench4"
SYST:COMM:LAN:APPL
SYST:COMM:LAN:HOST?
-> "lps-bench4"
```

Related commands

SYSTem:COMMunicate:LAN:HOSTname:STATus?, SYSTem:COMMunicate:LAN:DESCription, SYSTem:COMMunicate:LAN:APPLy

15.8 SYSTem:COMMunicate:LAN:HOSTname:STATus?

Returns the hostname currently used for mDNS advertisement.

The advertised host label is derived from the applied hostname; a collision on the link forces renaming, and a numeric suffix may then be appended. Without a collision the response is identical to LAN:HOSTname?. The returned value is the host label alone, without the .local suffix. The query reports the state of the running interface and is independent of staged values that have not been applied.

QUERY

```
SYSTem:COMMunicate:LAN:HOSTname:STATus?
```

Returns

<String>	The host label currently advertised through mDNS, without the .local suffix, in quotes.

Example

```
SYST:COMM:LAN:HOST?
-> "lps-bench4"
SYST:COMM:LAN:HOST:STAT?
-> "lps-bench4-2"
```

Related commands

SYSTem:COMMunicate:LAN:HOSTname, SYSTem:COMMunicate:LAN:DESCription:STATus?, SYSTem:COMMunicate:LAN:APPLy

15.9 SYSTem:COMMunicate:LAN:IPV4:ADDRess

Sets or queries the static IPv4 address of the Ethernet interface.

The address is given in dotted-decimal notation as a quoted string and is used in IPV4:MODE MANual. Setting it does not select MANual automatically and has no effect until LAN:APPLy. The query returns the staged static address, NONE when no manual address is configured; it is never the address obtained from DHCP or AutoIP, which is read with IPV4:STATus:ADDRess? and can be completely different in AUTO mode. An empty string clears the staged static value; the query then returns NONE.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV4:ADDRess
  <address>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<address>	String		Static IPv4 address in dotted-decimal notation, in quotes, for example "192.168.1.50".

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:ADDRess?
```

Returns

<String> or <CRD>	The staged static IPv4 address in quotes, or NONE when no manual address is configured.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	The string is not a syntactically valid IPv4 address. The previous staged address is unchanged.

Example

```
SYST:COMM:LAN:IPV4:ADDR ”192.168.1.50”
SYST:COMM:LAN:IPV4:ADDR?
-> ”192.168.1.50”
SYST:COMM:LAN:IPV4:MODE MAN
SYST:COMM:LAN:APPL
SYST:COMM:LAN:IPV4:STAT:ADDR?
-> ”192.168.1.50”
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:IPV4:SUBNetmask, SYSTem:COMMunicate:LAN:IPV4:GATeway, SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?

15.10 SYSTem:COMMunicate:LAN:IPV4:GATeway

Sets or queries the static IPv4 default gateway of the Ethernet interface.

The gateway is given in dotted-decimal notation as a quoted string and is used in IPV4:MODE MANual after LAN:APPLy. Setting it writes the staged LAN configuration only and does not select MANual. The query returns the staged gateway, NONE when none is configured; the gateway actually in use, which in AUTO mode comes from DHCP, is read with IPV4:STATus:GATeway?. An empty string clears the staged static value; the query then returns NONE.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV4:GATeway
  <address>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<address>	String		IPv4 default gateway in dotted-decimal notation, in quotes, for example ”192.168.1.1”.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:GATeway?
```

Returns

`<String> or <CRD>`	The staged IPv4 gateway in quotes, or NONE when none is configured.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	The string is not a syntactically valid IPv4 address. The previous staged gateway is unchanged.

Example

```
SYST:COMM:LAN:IPV4:GAT "192.168.1.1"
SYST:COMM:LAN:IPV4:GAT?
-> "192.168.1.1"
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:IPV4:ADDRess, SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway?, SYSTem:COMMunicate:LAN:APPLy

15.11 SYSTem:COMMunicate:LAN:IPV4:MODE

Selects or queries the IPv4 operating mode of the Ethernet interface.

AUTO obtains the address by DHCP and falls back to IPv4 link-local addressing when no DHCP address can be obtained; MANual uses the staged static address, subnet mask and gateway; DISabled switches IPv4 off. The mode is staged and takes effect with LAN:APPLy. IPv6 has its own independent mode, so the instrument can run either protocol alone or both together. The query returns the staged mode, not the method that supplied the address currently in use, which is read with IPV4:STATus?.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV4:MODE
  DISabled|AUTO|MANual
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
DISabled	CPD		IPv4 is switched off after the next LAN:APPLy.
AUTO	CPD		DHCP with IPv4 link-local (AutoIP) fallback. Factory and LAN-reset default.
MANual	CPD		Use the staged static address, subnet mask and gateway.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:MODE?
```

Returns

<CRD>	DIS, AUTO or MAN for the staged configuration.

Example

```
SYST:COMM:LAN:IPV4:MODE MAN
SYST:COMM:LAN:IPV4:MODE?
-> MAN
SYST:COMM:LAN:APPL
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:ADDRess, SYSTem:COMMunicate:LAN:IPV4:STATus?, SYSTem:COMMunicate:LAN:IPV6:MODE, SYSTem:COMMunicate:LAN:APPLy

15.12 SYSTem:COMMunicate:LAN:IPV4:STATus?

Returns the active IPv4 state of the Ethernet interface.

The response describes the interface as it is running, independently of staged values that have not been applied. DISABLED means that the applied IPv4 mode is DISabled, DOWN that the Ethernet link is down, and CONFIGURING that the link is up while address acquisition, DHCP, the AutoIP fallback or duplicate-address probing of a static address, is in progress. STATIC, DHCP and AUTOIP identify the method that supplied the currently valid address, so they also show whether AUTO mode reached a DHCP server or fell back to link-local addressing. CONFLICT reports an address conflict detected for the configured static address, LOST the loss of a previously valid dynamic address. Reading the state changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:STATus?
```

Returns

<CRD>	DISABLED, DOWN, CONFIGURING, STATIC, DHCP, AUTOIP, CONFLICT or LOST.

Example

```
SYST:COMM:LAN:IPV4:STAT?
-> DHCP
SYST:COMM:LAN:IPV4:STAT:ADDR?
-> "192.168.1.117"
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?, SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:IPV6:STATus?

15.13 SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?

Returns the IPv4 address currently assigned to the Ethernet interface.

The response is the address actually in use, whether it came from the static configuration, from DHCP or from AutoIP; NONE is returned when no IPv4 address is active. This query and IPV4:ADDRess? are deliberately separate: the latter reports the configured manual value, which in AUTO mode can be completely different or absent. The response reflects the applied configuration only and ignores staged changes. Reading it changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?
```

Returns

<String> or <CRD>	The active IPv4 address in dotted-decimal notation, in quotes, or NONE when no IPv4 address is active.

Example

```
SYST:COMM:LAN:IPV4:STAT:ADDR?
-> "192.168.1.117"
SYST:COMM:LAN:IPV4:ADDR?
-> NONE
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:STATus?, SYSTem:COMMunicate:LAN:IPV4:ADDRess, SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask?

15.14 SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway?

Returns the IPv4 default gateway currently in use by the Ethernet interface.

The response is the gateway actually active, whether it was configured statically or supplied by DHCP; NONE is returned when no gateway is active. The configured manual gateway is read with IPV4:GATeway? and can differ. The response reflects the applied configuration only and ignores staged changes. Reading it changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway?
```

Returns

<String> or <CRD>	The active IPv4 gateway in dotted-decimal notation, in quotes, or NONE when none is active.

Example

```
SYST:COMM:LAN:IPV4:STAT:GAT?
-> "192.168.1.1"
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:GATeway, SYSTem:COMMunicate:LAN:IPV4:STATus?, SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?

15.15 SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask?

Returns the IPv4 subnet mask currently in use by the Ethernet interface.

The response is the mask actually active, whether it was configured statically or supplied by DHCP or AutoIP; NONE is returned when no IPv4 configuration is active. The configured manual mask is read with IPV4:SUBNetmask? and can differ. The response reflects the applied configuration only and ignores staged changes. Reading it changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask?
```

Returns

`<String> or <CRD>`	The active subnet mask in dotted-decimal notation, in quotes, or `NONE` when none is active.

Example

```
SYST:COMM:LAN:IPV4:STAT:SUBN?
-> "255.255.255.0"
```

Related commands

`SYSTem:COMMunicate:LAN:IPV4:SUBNetmask`, `SYSTem:COMMunicate:LAN:IPV4:STATus?`, `SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?`

15.16 SYSTem:COMMunicate:LAN:IPV4:SUBNetmask

Sets or queries the static IPv4 subnet mask of the Ethernet interface.

The mask is given in dotted-decimal notation as a quoted string and is used in `IPV4:MODE MANual` after `LAN:APPLy`. Setting it writes the staged LAN configuration only and does not select `MANual`. The query returns the staged mask, `NONE` when none is configured; the mask actually in use is read with `IPV4:STATus:SUBNetmask?`.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV4:SUBNetmask
  <mask>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<mask>`	String		Subnet mask in dotted-decimal notation, in quotes, for example `"255.255.255.0"`.

QUERY

```
SYSTem:COMMunicate:LAN:IPV4:SUBNetmask?
```

Returns

`<String> or <CRD>`	The staged subnet mask in quotes, or `NONE` when none is configured.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	The string is not a syntactically valid IPv4 netmask. The previous staged mask is unchanged.

Example

```
SYST:COMM:LAN:IPV4:SUBN "255.255.255.0"
SYST:COMM:LAN:IPV4:SUBN?
-> "255.255.255.0"
```

Related commands

SYSTem:COMMunicate:LAN:IPV4:ADDRess, SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask?, SYSTem:COMMunicate:LAN:APPLy

15.17 SYSTem:COMMunicate:LAN:IPV6:ADDRess

Sets or queries the static IPv6 address of the Ethernet interface.

The address is given as a quoted string in normal IPv6 notation and is used in IPV6:MODE MANual together with IPV6:PREFix and IPV6:GATeway. Setting it does not change the mode and has no effect until LAN:APPLy. The query returns the staged static address, NONE when no manual address is configured; the address usable at the time of the query is read with IPV6:STATus:GLOBal?. An empty string clears the staged static value; the query then returns NONE.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV6:ADDRess
  <address>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<address>	String		Static IPv6 address in quotes, for example "2001:db8::10".

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:ADDRess?
```

Returns

<String> or <CRD>	The staged static IPv6 address in quotes, or NONE when no manual address is configured.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	The string is not a syntactically valid IPv6 address. The previous staged address is unchanged.

Example

```
SYST:COMM:LAN:IPV6:ADDR "2001:db8::10"
SYST:COMM:LAN:IPV6:PREF 64
SYST:COMM:LAN:IPV6:MODE MAN
SYST:COMM:LAN:APPL
SYST:COMM:LAN:IPV6:ADDR?
-> "2001:db8::10"
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:PREFix, SYSTem:COMMunicate:LAN:IPV6:MODE, SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?, SYSTem:COMMunicate:LAN:APPLy

15.18 SYSTem:COMMunicate:LAN:IPV6:GATeway

Sets or queries the static IPv6 gateway of the Ethernet interface.

The gateway, the router used in manual mode, is given as a quoted string in normal IPv6 notation. Setting it writes the staged LAN configuration only, does not change the mode and has no effect until LAN:APPLy. The query returns the staged gateway, NONE when none is configured. An empty string clears the staged static value; the query then returns NONE.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV6:GATeway
  <address>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<address>	String		IPv6 gateway or router address in quotes, for example "fe80::1".

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:GATeway?
```

Returns

`<String> or <CRD>`	The staged IPv6 gateway in quotes, or NONE when none is configured.

Errors

CODE	MESSAGE	CONDITION
-224	"Illegal parameter value"	The string is not a syntactically valid IPv6 address. The previous staged gateway is unchanged.

Example

```
SYST:COMM:LAN:IPV6:GAT "fe80::1"
SYST:COMM:LAN:IPV6:GAT?
-> "fe80::1"
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:ADDRess, SYSTem:COMMunicate:LAN:IPV6:MODE, SYSTem:COMMunicate:LAN:APPLy

15.19 SYSTem:COMMunicate:LAN:IPV6:MODE

Selects or queries the IPv6 operating mode of the Ethernet interface.

AUTO selects automatic IPv6 configuration; the mechanism used internally to obtain the usable global address is not part of the SCPI contract. MANual uses the staged static address, prefix length and gateway, DISabled switches IPv6 off. The mode is staged and takes effect with LAN:APPLy. IPv4 has its own independent mode. The query returns the staged mode, not the state of the running interface, which is read with IPV6:STATus?.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV6:MODE
  DISabled|AUTO|MANual
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
DISabled	CPD		IPv6 is switched off after the next LAN:APPLy.
AUTO	CPD		Automatic IPv6 configuration. Factory and LAN-reset default.
MANual	CPD		Use the staged static address, prefix length and gateway.

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:MODE?
```

Returns

<CRD> DIS, AUTO or MAN for the staged configuration.

Example

```
SYST:COMM:LAN:IPV6:MODE AUTO
SYST:COMM:LAN:IPV6:MODE?
-> AUTO
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:ADDRess, SYSTem:COMMunicate:LAN:IPV6:STATus?, SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:APPLy

15.20 SYSTem:COMMunicate:LAN:IPV6:PREFix

Sets or queries the prefix length of the static IPv6 address.

The prefix length is given in bits and belongs to the staged static IPv6 address used in IPV6:MODE MANual. Setting it writes the staged LAN configuration only and takes effect with LAN:APPLy. The query returns the staged value.

COMMAND

```
SYSTem:COMMunicate:LAN:IPV6:PREFix
  <bits>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<bits>	NR1	0 to 128	Prefix length of the static IPv6 address in bits.

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:PREFix?
```

Returns

<NR1> The staged prefix length in bits.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<bits> is outside 0 to 128. The previous staged prefix length is unchanged.

Example

```
SYST:COMM:LAN:IPV6:PREF 64
SYST:COMM:LAN:IPV6:PREF?
-> 64
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:ADDRess, SYSTem:COMMunicate:LAN:IPV6:MODE, SYSTem:COMMunicate:LAN:APPLy

15.21 SYSTem:COMMunicate:LAN:IPV6:STATus?

Returns the active IPv6 state of the Ethernet interface.

The response describes the interface as it is running, independently of staged values that have not been applied. DISABLED means that the applied IPv6 mode is DISabled, DOWN that the Ethernet link is down, and CONFIGURING that the link is up but no global address is valid yet, because only the link-local address exists or duplicate-address detection or router discovery is still in progress. STATIC means that a valid configured static global address is active, DYNAMIC that a valid automatically configured global address is active. CONFLICT reports a failed duplicate-address detection for the configured static address, LOST that a previously valid dynamic global address has expired without replacement. Reading the state changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:STATus?
```

Returns

<CRD>	DISABLED, DOWN, CONFIGURING, STATIC, DYNAMIC, CONFLICT or LOST.

Example

```
SYST:COMM:LAN:IPV6:STAT?
-> DYNAMIC
SYST:COMM:LAN:IPV6:STAT:GLOB?
-> "2001:db8::7a3f"
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?,
SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal?,
SYSTem:COMMunicate:LAN:IPV6:MODE

15.22 SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?

Returns the active non-link-local IPv6 address of the Ethernet interface.

The response is the address actually usable at the time of the query, whether it was configured statically or obtained automatically; NONE is returned when no non-link-local address is valid. When more than one global or unique-local address is present, the instrument selects the preferred usable one. The configured static value is read with IPV6:ADDRess? and can differ. Reading the address changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?
```

Returns

<String> or <CRD>	The active non-link-local IPv6 address in quotes, or NONE when none is valid.

Example

```
SYST:COMM:LAN:IPV6:STAT:GLOB?
-> "2001:db8::7a3f"
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:STATus?,
SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal?,
SYSTem:COMMunicate:LAN:IPV6:ADDRess

15.23 SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal?

Returns the active IPv6 link-local address of the Ethernet interface.

The link-local address exists whenever IPv6 is up and is independent of the mode and of any global address; NONE is returned when no valid link-local address exists. Reading the address changes no communication state and does not depend on staged values.

QUERY

```
SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal?
```

Returns

<String> or <CRD>	The active IPv6 link-local address in quotes, or NONE when none is valid.

Example

```
SYST:COMM:LAN:IPV6:STAT:LINK?
-> "fe80::204:a3ff:fe12:3456"
```

Related commands

SYSTem:COMMunicate:LAN:IPV6:STATus?,
SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?,
SYSTem:COMMunicate:LAN:MACaddress?

15.24 SYSTem:COMMunicate:LAN:MACaddress?

Returns the Ethernet MAC address of the instrument.

The address is returned as a colon-separated hexadecimal string, for example `00:04:A3:12:34:56`. It is a fixed hardware property and is not user-programmable through the SCPI interface; there is no setting form. The response is independent of the LAN configuration, of the staged values and of the link state. Reading it changes no communication state.

QUERY

```
SYSTem:COMMunicate:LAN:MACaddress?
```

Returns

`<String>`	The Ethernet MAC address as six colon-separated hexadecimal bytes, in quotes.

Example

```
SYST:COMM:LAN:MAC?
-> "00:04:A3:12:34:56"
```

Related commands

`SYSTem:COMMunicate:LAN:HOSTname:STATus?`, `SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?`, `*IDN?`

15.25 SYSTem:COMMunicate:LAN:RESet

Restores the LAN configuration to the product defaults and applies them immediately.

The defaults are IPv4 `AUTO`, IPv6 `AUTO`, the product-generated default hostname and service description, the manual IPv4 address and gateway `NONE` with the subnet mask `255.255.255.0`, and the manual IPv6 address and gateway `NONE` with the prefix length 64. They are stored as the new persistent LAN configuration and applied at once, so no separate `LAN:APPLy` is needed and staged changes are dropped with them. The command affects the LAN configuration only: the USB class, the RS-232 settings, the LXI event domain and the instrument configuration are untouched, and it is not equivalent to *RST. Because the addressing changes, the current LAN session may be lost.

COMMAND

```
SYSTem:COMMunicate:LAN:RESet
```

Example

```
SYST:COMM:LAN:RES
SYST:COMM:LAN:IPV4:MODE?
-> AUTO
SYST:COMM:LAN:IPV4:ADDR?
-> NONE
```

Related commands

SYSTem:COMMunicate:LAN:APPLy, SYSTem:COMMunicate:LAN:DISCard, SYSTem:COMMunicate:LAN:IPV4:MODE, SYSTem:COMMunicate:LAN:HOSTname

15.26 SYSTem:COMMunicate:LXI:EVENt:DOMain

Sets or queries the LXI Event Messaging domain of the instrument.

The domain byte separates independent test systems that share one LAN: received LXI event messages whose domain does not match are ignored, and transmitted messages are tagged with it once transmit support exists. This is the LXI Event Messaging domain, not the IEEE 1588 / PTP domain number; the two are independent settings.

COMMAND

```
SYSTem:COMMunicate:LXI:EVENt:DOMain
  <domain>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<domain>	NR1	0 to 255	LXI Event Messaging domain byte. Factory default 0.

QUERY

```
SYSTem:COMMunicate:LXI:EVENt:DOMain?
```

Returns

<NR1>	The configured domain, 0 to 255.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<domain> is outside 0 to 255. The domain is unchanged.
-221	"Settings conflict"	An operation whose trigger source is LXI is armed or active. The domain is unchanged.

Example

```
SYST:COMM:LXI:EVEN:DOM 3
SYST:COMM:LXI:EVEN:DOM?
-> 3
```

Related commands

TRIGger:LXI:EVENt, TRIGger[:SEQuence]:SOURce, SYSTem:TIME:PTP:STATus?

15.27 SYSTem:COMMunicate:RLSTate

Sets or queries the local/remote state of the instrument as a whole.

The instrument has one global control state with the values LOC, REM and RWL; SYSTem:LOCal, SYSTem:REMote and SYSTem:RWLock set the same state, it is not maintained per interface, and the most recently applied transition is the effective one. In LOC the front panel is fully usable and the remote interfaces stay available; in REM local editing is inhibited but the user may return to local operation from the front panel without losing the remote connection; in RWL that local takeover is disabled until the lockout is released remotely or by a transport-level Go-To-Local operation. Changing the state modifies no source, measurement, protection or acquisition setting, and physical power controls and safety functions are never disabled. The state neither acquires nor releases a HiSLIP or VXI-11 transport lock, which is reported separately by SYSTem:LOCK:OWNer?.

COMMAND

```
SYSTem:COMMunicate:RLSTate
  LOC|REM|RWL
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
LOC	CPD		Local. Normal front-panel operation, remote interfaces remain reachable.
REM	CPD		Remote. Local editing is inhibited; the user may return to local operation from the front panel.
RWL	CPD		Remote with local lockout. Local takeover through the normal GUI is disabled.

QUERY

```
SYSTem:COMMunicate:RLSTate?
```

Returns

<CRD>	LOC, REM or RWL, the current global control state.

Example

```
SYST:COMM:RLST RWL
SYST:COMM:RLST?
-> RWL
SYST:LOC
SYST:COMM:RLST?
-> LOC
```

Related commands

SYSTem:LOCal, SYSTem:REMote, SYSTem:RWLock, SYSTem:LOCK:OWNer?

15.28 SYSTem:COMMunicate:SELF?

Returns the communication interface through which the query itself was received.

The response names the transport of the calling session; for network transports connection information such as the remote IP address may be appended. The current interface names are USB, RS232, HiSLIP, VXI-11 and SOCKET. The query is intended for diagnostics and for software that can reach the instrument over several transports at the same time, and it says nothing about which interface holds a transport lock, which is reported by SYSTem:LOCK:OWNer?. The query changes no communication state.

QUERY

```
SYSTem:COMMunicate:SELF?
```

Returns

<String>	The interface name of the calling session, optionally followed by connection information, in quotes.

Example

```
SYST:COMM:SELF?
-> "HiSLIP 192.168.1.20"
```

Related commands

SYSTem:LOCK:OWNer?, SYSTem:COMMunicate:USB:CLASs, SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?

15.29 SYSTem:COMMunicate:SERial:BAUDrate

Sets or queries the baud rate of the RS-232 interface.

One of nine fixed baud rates is selected; the framing is fixed at 8N1, 8 data bits, no parity, one stop bit, and is not programmable. The change is applied immediately, not at the next connection: when the command is issued over RS-232 itself, all further communication must use the new baud rate.

COMMAND

```
SYSTem:COMMunicate:SERial:BAUDrate
  <baud>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<baud>	NR1, bit/s	300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200	New baud rate. Factory default 115200.

QUERY

```
SYSTem:COMMunicate:SERial:BAUDrate?
```

Returns

<NR1> The configured baud rate in bit/s.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<baud> is not one of the nine supported baud rates. The baud rate is unchanged.

Example

```
SYST:COMM:SER:BAUD 9600
SYST:COMM:SER:BAUD?
-> 9600
```

Related commands

SYSTem:COMMunicate:SERial:FLOWcontrol, SYSTem:COMMunicate:SELF?, SYSTem:COMMunicate:USB:CLASs

15.30 SYSTem:COMMunicate:SERial:FLOWcontrol

Selects or queries the flow control of the RS-232 interface.

NONE disables flow control, RTSCTS enables RTS/CTS hardware handshake; software flow control with XON/XOFF is not provided. The change is applied immediately, not at the next connection.

COMMAND

```
SYSTem:COMMunicate:SERial:FLOWcontrol
  NONE|RTSCTS
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
NONE	CPD		No flow control. Factory default.
RTSCTS	CPD		RTS/CTS hardware flow control. A fixed token without a short form, sent and returned exactly as RTSCTS.

QUERY

```
SYSTem:COMMunicate:SERial:FLOWcontrol?
```

Returns

<CRD>	NONE or RTSCTS.

Example

```
SYST:COMM:SER:FLOW RTSCTS
SYST:COMM:SER:FLOW?
-> RTSCTS
```

Related commands

SYSTem:COMMunicate:SERial:BAUDrate, SYSTem:COMMunicate:SELF?

15.31 SYSTem:COMMunicate:USB:CLASs

Selects or queries the USB device class used for remote control.

The two classes are mutually exclusive: TMC presents the instrument as a USB Test and Measurement device, CDC as a virtual serial port. The new class is applied immediately by disconnecting, reconfiguring and re-enumerating the USB device; when the command arrives over the USB session itself, that session terminates as the device re-enumerates, and the controller has to open the new device. LAN and RS-232 sessions are not affected, and the setting needs no LAN:APPLy.

COMMAND

```
SYSTem:COMMunicate:USB:CLASs
  CDC|TMC
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
CDC	CPD		USB Communications Device Class, a virtual serial interface.
TMC	CPD		USB Test and Measurement Class. Factory default.

QUERY

```
SYSTem:COMMunicate:USB:CLASs?
```

Returns

<CRD>	CDC or TMC.

Example

```
SYST:COMM:USB:CLAS?
-> TMC
SYST:COMM:USB:CLAS CDC
```

Related commands

SYSTem:COMMunicate:SELF?, SYSTem:COMMunicate:SERial:BAUDrate, *IDN?

15.32 SYSTem:CSV:DECimal

Selects or queries the decimal separator used in CSV files of the instrument.

The separator applies to the numerical values of CSV files the instrument writes and of CSV files it reads, and exists for interoperability with regional spreadsheet conventions. It never changes SCPI numeric syntax: program messages and ASCII query responses keep the locale-independent decimal point and the comma as parameter separator. The selection is not coupled to CSV:DELimiter; choosing the comma for both is allowed, and producing a file that can still be parsed is then the responsibility of the program. A data log snapshots all three CSV settings when INITiate DLOG starts an ASCII log, and a sequence file selected with [SOURce:]ARB:SEQuence:FILE is read with the delimiter and the decimal separator in effect at INITiate, when it is loaded; selecting the path records no CSV setting, and a later change never affects a running log or a sequence that is loading or playing.

COMMAND

```
SYSTem:CSV:DECimal
  DOT|COMMa
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
DOT	CPD		Decimal point . in CSV values. Factory default.
COMMa	CPD		Decimal comma , in CSV values.

QUERY

```
SYSTem:CSV:DECimal?
```

Returns

<CRD>	DOT or COMM.

Example

```
SYST:CSV:DEC COMM
SYST:CSV:DEL SEM
SYST:CSV:DEC?
-> COMM
```

Related commands

SYSTem:CSV:DELimiter, SYSTem:CSV:LINeending, FORMat[:DATA], INITiate

15.33 SYSTem:CSV:DELimiter

Selects or queries the field delimiter used in CSV files of the instrument.

The delimiter separates the columns of CSV files the instrument writes and of CSV files it reads, such as a sequence file. It does not affect SCPI response separators: several query responses in one message stay separated by semicolons and list responses by commas, independently of this setting. The selection is not coupled to CSV:DECimal. A data log snapshots all three CSV settings when INITiate DLOG starts an ASCII log, and a sequence file is read with the delimiter and the decimal separator in effect at INITiate, when it is loaded; selecting the path with [SOURce:]ARB:SEQuence:FILE neither reads the file nor records a CSV setting, and a change during the load does not affect it.

COMMAND

```
SYSTem:CSV:DELimiter
  COMMa|SEMicolon
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
COMMa	CPD		Comma , as field delimiter. Factory default.
SEMicolon	CPD		Semicolon ; as field delimiter.

QUERY

```
SYSTem:CSV:DELimiter?
```

Returns

<CRD>	COMM or SEM.

Example

```
SYST:CSV:DEL SEM
SYST:CSV:DEL?
-> SEM
```

Related commands

SYSTem:CSV:DECimal, SYSTem:CSV:LINeending, [SOURce:]ARB:SEQuence:FILE, INITiate

15.34 SYSTem:CSV:LINeending

Selects or queries the line ending written into CSV files of the instrument.

The choice applies to every line of a generated CSV file: the comment and metadata header, the column-header line and all data rows. It controls generated output only; CSV input is always accepted with either LF or CRLF line endings, so a file written on a PC is read regardless of this setting. SCPI responses are unaffected and keep their own message terminator. A data log snapshots all three CSV settings when INITiate DLOG starts an ASCII log, so a change during a running log affects only later CSV-producing operations.

COMMAND

```
SYSTem:CSV:LINeending
  LF|CRLF
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
LF	CPD		Line feed only. Factory default.
CRLF	CPD		Carriage return followed by line feed.

QUERY

```
SYSTem:CSV:LINeending?
```

Returns

`<CRD>`	LF or CRLF.

Example

```
SYST:CSV:LIN CRLF
SYST:CSV:LIN?
-> CRLF
```

Related commands

`SYSTem:CSV:DELimiter`, `SYSTem:CSV:DECimal`, `FORMat[:DATA]`

15.35 SYSTem:DATE

Sets or queries the local civil calendar date of the instrument.

The date is kept by the battery-backed real-time clock, which is the normal time source of the instrument and works without a network connection; it feeds the timestamps of logs and events and the absolute-time trigger `TRIGger:TIME`. Setting the date preserves the current time of day, and only future timestamps are affected: data already recorded is never rewritten. While a validly synchronized PTP clock is the authoritative time source, that is, while `SYSTem:TIME:SOURce?` returns PTP, the date cannot be written; the query works in either case. A step that moves the clock past an armed `TRIGger:TIME` target produces the time event at once, exactly once; a step back before the target does not produce it again.

COMMAND

```
SYSTem:DATE
  <year>,<month>,<day>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<year>`	NR1	2000 to 2099	Local civil year.
`<month>`	NR1	1 to 12	Local civil month.
`<day>`	NR1	1 to 31	Local civil day. Must exist in the selected month and year.

QUERY

```
SYSTem:DATE?
```

Returns

<NR1>,<NR1>,<NR1>	The current local civil date as year, month and day.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A validly synchronized PTP clock is the authoritative time source. The clock is unchanged.
-222	"Data out of range"	<year> is outside 2000 to 2099, <month> is outside 1 to 12, or <day> does not exist in the selected month and year. The clock is unchanged.

Example

```
SYST:DATE 2026,9,9
SYST:DATE?
-> 2026,9,9
SYST:TIME:SOUR?
-> RTC
```

Related commands

SYSTem:TIME, SYSTem:TZONe, SYSTem:TIME:SOURce?, TRIGger:TIME

15.36 SYSTem:ERRor:ALL?

Returns and removes all entries of the error queue of the calling session.

Every unread entry is returned oldest first as a comma-separated list of <code>,"<message>" pairs, and the queue is emptied; an empty queue returns 0,"No error". The query reads the same per-context queue as SYSTem:ERRor[:NEXT]? and consumes nothing from another session. Only the error queue is affected: the Standard Event register and the other status registers keep their contents, and the EAV bit of this context clears. The response is text and is not affected by FORMat:SREGister.

QUERY

```
SYSTem:ERRor:ALL?
```

Returns

`<NR1>,<String>[,<NR1>,<String>...]` All entries of the queue, oldest first, each as the error code and the message in quotes, or `0,"No error"` when the queue is empty.

Example

```
VOLT 1E9
SYST:COMM:LAN:IPV4:ADDR "1.2.3"
SYST:ERR:ALL?
-> -222,"Data out of range",-224,"Illegal parameter value"
SYST:ERR:ALL?
-> 0,"No error"
```

Related commands

`SYSTem:ERRor[:NEXT]?`, `SYSTem:ERRor:CODE:ALL?`, `SYSTem:ERRor:COUNt?`, `*CLS`

15.37 SYSTem:ERRor:CLEar

Removes all entries from the error queue of the calling session.

Every session has its own error queue, and the command empties only the queue of the calling context; the queues of other contexts are untouched, as is any error a concurrent controller has not read yet. The Standard Event, Operation, Questionable and Measurement registers are not cleared, no condition register changes and no instrument setting or running operation is affected; `*CLS` is the broader command that clears the event registers and the error queue of the calling context together. After the command `SYSTem:ERRor:COUNt?` returns `0` and the EAV bit of the Status Byte of this context is clear. Clearing an already empty queue is harmless.

COMMAND

```
SYSTem:ERRor:CLEar
```

Example

```
SYST:ERR:COUN?
-> 3
SYST:ERR:CLE
SYST:ERR:COUN?
-> 0
```

Related commands

SYSTem:ERRor[:NEXT]?, SYSTem:ERRor:COUNt?, *CLS

15.38 SYSTem:ERRor:CODE:ALL?

Returns and removes the codes of all entries of the error queue of the calling session.

The query behaves like SYSTem:ERRor:ALL? but returns only the codes as a comma-separated list of NR1 values, oldest first, and empties the queue; an empty queue returns 0. The messages are discarded together with the entries and cannot be read afterwards. Only the queue of the calling session is affected, and the EAV bit of this context clears.

QUERY

```
SYSTem:ERRor:CODE:ALL?
```

Returns

<NR1>[,<NR1>...]	The codes of all entries of the queue, oldest first, or 0 when the queue is empty.

Example

```
VOLT 1E9
SYST:COMM:LAN:IPV4:ADDR "1.2.3"
SYST:ERR:CODE:ALL?
-> -222,-224
SYST:ERR:CODE:ALL?
-> 0
```

Related commands

SYSTem:ERRor:ALL?, SYSTem:ERRor:CODE[:NEXT]?, SYSTem:ERRor:COUNt?, *CLS

15.39 SYSTem:ERRor:CODE[:NEXT]?

Returns and removes the code of the oldest entry of the error queue of the calling session.

The query behaves like SYSTem:ERRor[:NEXT]? but returns only the NR1 code of the oldest entry and removes that entry; an empty queue returns 0. The message is discarded together with the entry and cannot be read afterwards. Only the queue of the calling session is affected; reading the last entry clears the EAV bit of this context.

QUERY

```
SYSTem:ERRor:CODE[:NEXT]?
```

Returns

<NR1>	The error code of the oldest entry, or 0 when the queue of the calling session is empty.

Example

```
VOLT 1E9
SYST:ERR:CODE?
-> -222
SYST:ERR:CODE?
-> 0
```

Related commands

SYSTem:ERRor[:NEXT]?, SYSTem:ERRor:CODE:ALL?, SYSTem:ERRor:COUNt?, *CLS

15.40 SYSTem:ERRor:COUNt?

Returns the number of unread entries in the error queue of the calling session.

The query is non-destructive: it reports how many entries SYSTem:ERRor[:NEXT]? would still return, without removing any of them and without changing the Status Byte. It counts only the queue of the calling session; errors queued in another context are not included. A response of 0 means that the context has no pending error, which is the same state in which SYSTem:ERRor[:NEXT]? returns 0,"No error". The query itself never queues an error.

QUERY

```
SYSTem:ERRor:COUNt?
```

Returns

<NR1>	The number of unread entries in the error queue of the calling session.

Example

```
SYST:ERR:COUN?
-> 2
SYST:ERR?
-> -222,"Data out of range"
SYST:ERR:COUN?
-> 1
```

Related commands

`SYSTem:ERRor[:NEXT]?`, `SYSTem:ERRor:CLEar`, `*CLS`

15.41 SYSTem:ERRor[:NEXT]?

Returns and removes the oldest entry from the error queue of the calling session.

Entries are returned first in, first out as `<code>,"<message>"`; when the queue is empty the response is `0,"No error"`. Every interface instance owns one session with its own queue of at least 16 unread entries; a new connection to that instance takes the context over and does not clear it. Parser, parameter, query and execution errors caused by a program message are queued in the context that sent it, and the asynchronous errors of an operation in the context that started it, even when another controller has taken over that interface instance since; reading here consumes nothing from another context. The query affects only the error queue; the Standard Event register and the other status registers keep their contents, and the EAV bit of this context clears when its queue runs empty. When the queue is full, its newest entry is replaced by `-350,"Queue overflow"` and the older entries are kept.

QUERY

```
SYSTem:ERRor[:NEXT]?
```

Returns

`<NR1>,<String>`	The error code and the error message in quotes, or `0,"No error"` when the queue of the calling session is empty.

Example

```
VOLT 1E6
SYST:ERR?
-> -222,"Data out of range"
SYST:ERR?
-> 0,"No error"
```

Related commands

`SYSTem:ERRor:COUNt?`, `SYSTem:ERRor:CLEar`, `*CLS`, `*ESR?`

15.42 SYSTem:LFRequency

Sets or queries the mains frequency of the installation site.

The value tells the instrument which mains period to use for functions that average over the mains period or reject mains hum. It is an installation setting, not a measurement control: it does not change the acquisition interval `SENSe:SWEep:TINTerval`, the raw acquisition timing or the scalar one-point measurement semantics, and the instrument exposes no NPLC model. The query returns the configured value, or the factory default with the selector `DEFault`.

COMMAND

```
SYSTem:LFRequency
  50|60|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`50 | 60`	NR1, Hz	50 or 60	Mains frequency at the installation site.
`DEFault`	CPD		Selects the factory default 50 Hz.

QUERY

```
SYSTem:LFRequency?
  [DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`DEFault`	CPD		Returns the factory default 50 instead of the configured value. Omitted: the configured value is returned.

Returns

`<NR1>, Hz`	The configured mains frequency, `50` or `60`.

Errors

CODE	MESSAGE	CONDITION
-222	`"Data out of range"`	The value is neither `50` nor `60`. The mains frequency is unchanged.

Example

```
SYST:LFR 60
SYST:LFR?
-> 60
SYST:LFR? DEF
-> 50
```

Related commands

SENSe:SWEep:TINTerval, SYSTem:VERSion?

15.43 SYSTem:LOCal

Returns the instrument to local control.

The global control state becomes LOC, normal front-panel and touchscreen operation is restored, and an active local lockout is released, which is the remote way out of RWL. No source, measurement, protection or acquisition setting is changed and a running triggered operation continues; the remote connections stay open and can send further commands. The command does not release a HiSLIP or VXI-11 transport lock, which is independent of the local/remote state. Sending it while the instrument is already local is harmless.

COMMAND

```
SYSTem:LOCal
```

Example

```
SYST:RWL
SYST:LOC
SYST:COMM:RLST?
-> LOC
```

Related commands

SYSTem:REMote, SYSTem:RWLock, SYSTem:COMMunicate:RLSTate

15.44 SYSTem:LOCK:OWNer?

Returns the communication interface that currently holds a transport lock.

Shared and exclusive device locking is a transport function, HiSLIP locking and the VXI-11 `device_lock` operation; the instrument offers no SCPI command to acquire or release such a lock, and this query only reports who holds one. The response names the interface holding the lock, followed by connection information where the transport provides it, and is NONE while no lock is held. The query is passive from every interface, including one that is itself locked out: it never acquires, refreshes or releases a lock and never blocks. The lock state is unrelated to the local/remote state.

QUERY

```
SYSTem:LOCK:OWNer?
```

Returns

`<String> or <CRD>`	The interface holding the transport lock with its connection information, in quotes, or NONE while no lock is held.

Example

```
SYST:LOCK:OWN?
-> "HiSLIP 192.168.1.100"
SYST:LOCK:OWN?
-> NONE
```

Related commands

`SYSTem:COMMunicate:SELF?`, `SYSTem:COMMunicate:RLSTate`, `SYSTem:RWLock`

15.45 SYSTem:REMote

Puts the instrument under remote control.

The global control state becomes REM and normal local editing is inhibited while remote control is active. The local user may still request a return to local operation through the front panel; that transition does not terminate the remote connection and does not stop a running triggered operation. Entering remote changes no source, measurement, protection or acquisition setting and acquires no transport lock. The state is global, not per interface, so a later transition requested by another interface or transport replaces it.

COMMAND

```
SYSTem:REMote
```

Example

```
SYST:REM
SYST:COMM:RLST?
-> REM
```

Related commands

SYSTem:LOCal, SYSTem:RWLock, SYSTem:COMMunicate:RLSTate

15.46 SYSTem:RWLock

Puts the instrument under remote control with local lockout.

The global control state becomes RWL: the instrument is remote and normal front-panel and touchscreen control is locked, so the user cannot cancel remote operation through the normal GUI. The lockout is released by SYSTem:LOCal, by SYSTem:COMMunicate:RLSTate LOC or by an equivalent transport-level Go-To-Local operation. Physical power controls and safety functions that must remain available independently of software are not disabled. The command takes no argument, changes no instrument configuration and acquires no transport lock.

COMMAND

```
SYSTem:RWLock
```

Example

```
SYST:RWL
SYST:COMM:RLST?
-> RWL
```

Related commands

SYSTem:LOCal, SYSTem:REMote, SYSTem:COMMunicate:RLSTate, SYSTem:LOCK:OWNer?

15.47 SYSTem:TIME

Sets or queries the local civil time of day of the instrument.

The time of day is kept by the battery-backed real-time clock in 24-hour notation and is presented in local civil time, that is, absolute time shifted by SYSTem:TZONe. Setting it preserves the current date and affects only future timestamps of logs, events and the absolute-time trigger; recorded data is never rewritten. While a validly synchronized PTP clock is the authoritative time source, the time cannot be written, whereas the query keeps working; SYSTem:TIME:SOURce? states which of the two applies. A step that moves the clock past an armed TRIGger:TIME target produces the time event at once, exactly once; a step back before the target does not produce it again.

COMMAND

```
SYSTem:TIME
  <hour>,<minute>,<second>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<hour>	NR1	0 to 23	Local civil hour in 24-hour notation.
<minute>	NR1	0 to 59	Local civil minute.
<second>	NR1	0 to 59	Local civil second.

QUERY

```
SYSTem:TIME?
```

Returns

<NR1>,<NR1>,<NR1> The current local civil time as hour, minute and second in 24-hour notation.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	A validly synchronized PTP clock is the authoritative time source. The clock is unchanged.
-222	"Data out of range"	A field is outside its range. The clock is unchanged.

Example

```
SYST:TIME 14,32,5
SYST:TIME?
-> 14,32,5
SYST:DATE?
-> 2026,9,9
```

Related commands

SYSTem:DATE, SYSTem:TZONe, SYSTem:TIME:SOURce?, TRIGger:TIME

15.48 SYSTem:TIME:PTP:STATus?

Returns the synchronization state of the PTP clock.

The instrument provides IEEE 1588-2008 PTPv2 synchronization with hardware timestamping for the absolute-time functions that need it; the PTP profile itself is not configurable over SCPI, but its resulting state is observable here. DISABLED means the PTP service is not running, UNSYNCED that it runs without valid synchronized time, SYNCED that the clock is validly synchronized and the absolute time base is disciplined, and HOLDOVER that an established synchronization was lost and the clock continues free-running from its last disciplined state. Losing synchronization never invalidates the clock: the real-time clock carries on from the most recently established time, and SYSTem:DATE and SYSTem:TIME become writable again. The query is a read-only observation and configures nothing. SYNCED requires this instrument to be a PTP slave locked to a grandmaster with a valid UTC offset; an instrument acting as PTP master reports UNSYNCED, because its own clock is not disciplined by an external reference. SYSTem:TIME:SOURce? returns PTP exactly while the state is SYNCED.

QUERY

```
SYSTem:TIME:PTP:STATus?
```

Returns

<CRD>	DISABLED, UNSYNCED, SYNCED or HOLDOVER.

Example

```
SYST:TIME:PTP:STAT?
-> SYNCED
SYST:TIME:SOUR?
-> PTP
```

Related commands

SYSTem:TIME:SOURce?, SYSTem:DATE, SYSTem:TIME

15.49 SYSTem:TIME:SOURce?

Returns which time source currently disciplines the absolute time of the instrument.

RTC means the battery-backed real-time clock is authoritative, which is the normal case and does not require a network connection; PTP means a validly synchronized PTP clock is disciplining the absolute time base and is authoritative. The response therefore also states whether manual clock writes are currently accepted: SYSTem:DATE and SYSTem:TIME are accepted while RTC is returned and rejected with -221,"Settings conflict" while PTP is returned. A PTP-disciplined time may be written back into the real-time clock, so a corrected time survives a later power cycle. The query is read-only.

QUERY

```
SYSTem:TIME:SOURce?
```

Returns

<CRD>	RTC or PTP.

Example

```
SYST:TIME:SOUR?
-> RTC
SYST:TIME 14,32,5
```

Related commands

SYSTem:TIME:PTP:STATus?, SYSTem:DATE, SYSTem:TIME, SYSTem:TZONe

15.50 SYSTem:TZONe

Sets or queries the fixed local offset from UTC.

The offset is given in hours, positive east of UTC and negative west of it, and relates the local civil time shown by SYSTem:DATE and SYSTem:TIME to the absolute time of the instrument. Changing it changes only that presentation: the underlying absolute time, running operations and already recorded timestamps are not touched. The supported range is -14 to +14 and the value must be an exact multiple of 0.25 h, so 5.5 and 5.75 are accepted while 5.1 is not. The offset is a fixed value; automatic daylight-saving rules are not part of the command set, and a seasonal change is programmed by writing the new offset.

COMMAND

```
SYSTem:TZONe
  <offset>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<offset>	NRf, h	-14 to +14	Local offset from UTC in hours, in steps of 0.25 h. Positive is east of UTC. Factory default 0.

QUERY

```
SYSTem:TZONe?
```

Returns

<NR3>, h — The configured local offset from UTC in hours.

Errors

CODE	MESSAGE	CONDITION
-222	"Data out of range"	<offset> is outside -14 h to +14 h or is not a multiple of 0.25 h. The offset is unchanged.

Example

```
SYST:TZON 5.75
SYST:TZON?
-> 5.750000000E+00
SYST:TIME?
-> 20,17,5
```

Related commands

SYSTem:DATE, SYSTem:TIME, SYSTem:TIME:SOURce?

15.51 SYSTem:VERSion?

Returns the SCPI standard version the command set is based on.

The response is `1999.0` and identifies the level of the SCPI standard, not the firmware of the instrument; the firmware revision and the model are read with `*IDN?`, the installed options with `*OPT?`. The value is constant and does not change with the firmware version, the model or the interface used. The query has no side effects and queues no error.

QUERY

```
SYSTem:VERSion?
```

Returns

`<NR2>`	`1999.0`, the SCPI standard version of the command set.

Example

```
SYST:VERS?
-> 1999.0
```

Related commands

`*IDN?`, `*OPT?`

16

CHAPTER 16

TRACe commands

This chapter describes the two commands that act on the retained acquisition record in the internal sample memory: releasing it and exporting it to a file. The record itself is configured with the SENSe commands, armed with `INITiate` and read back with `FETCh:ARRay`; both commands here are events without a query form and are permitted only while the acquisition engine is idle.

IN THIS CHAPTER

16.1 `TRACe:CLEar` 353

16.2 `TRACe:SAVE` 354

16.1 TRACe:CLEar

Releases the retained acquisition record and resets its export cursor.

The command discards the retained record, the global consuming cursor and any still-unread terminal remainder, and returns the sample memory to its normal monitor layout; `FETCh:ARRay:POINts?` returns `0` afterwards and neither a record readback nor a streaming block is available. Acquisition function, target point count, interval, trigger configuration, running statistics and the transfer-format settings are unchanged. It is rejected while an operation with an ADC participant is armed or active, DLog included, and permitted after normal completion or `ABORt`. A new acquisition never needs it: a successful `INITiate` with an ADC participant discards the previous record and repartitions the buffer by itself. A DLog file that has already been written is never deleted or truncated by this command.

COMMAND

```
TRACe:CLEar
```

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active, or DLog owns the acquisition sink. The running acquisition and all captured data are unchanged.

Example

```
FETC:ARR:POIN?
-> 4096
TRAC:CLE
FETC:ARR:POIN?
-> 0
```

Related commands

TRACe:SAVE, FETCh:ARRay:POINts?, ABORt, INITiate

16.2 TRACe:SAVE

Writes the retained acquisition record to a file.

The export contains every point that is currently valid in the retained record; points already consumed by FETCh:ARRay:NEXT? are not resurrected. The operation is non-destructive: neither the record nor the consuming cursor is changed, and the same record can be exported again or fetched afterwards. FORMat[:DATA] and FORMat:BORDer, and for ASCII output the CSV preferences, determine the file representation; the file extension has no meaning, and the metadata fields follow the DLog conventions, with # First point: giving the global 1-based acquisition index of the first exported point. Missing parent directories are created and an existing target file is truncated; the command completes when the file has been flushed and closed. It is rejected while the acquisition engine is armed or measuring, DLog included, and a write failure leaves the partially written file in place without setting the STORAGE_ERROR status bit.

COMMAND

```
TRACe:SAVE
  "<path>"
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<path>	String		Path of the destination file, absolute and in quotes. The storage prefix USB: selects the external USB stick and is also what an unprefixed absolute path means; INT: is reserved for internal storage. / separates the elements, . and .. are not accepted.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation with an ADC participant is armed or active, or DLog owns the acquisition sink. No file is written.
-230	"Data corrupt or stale"	No valid retained acquisition point exists. No file is written.
-250	"Mass storage error"	The filesystem operation failed for another reason. The record is unchanged.
-251	"Missing mass storage"	The selected storage volume is not available on the instrument. No file is written.
-252	"Missing media"	No USB mass-storage medium is present. No file is written.
-253	"Corrupt media"	The medium is unusable. No file is written.
-254	"Media full"	The free space was insufficient. The partially written file stays in place; the record and the cursor are unchanged.
-257	"File name error"	The path is syntactically invalid or not supported by the storage. No file is written.
-258	"Media protected"	The medium or the path is write-protected. No file is written.

Example

```
INIT ACQ
*OPC?
-> 1
TRAC:SAVE "/logs/run1.csv"
FETC:ARR:POIN?
-> 4096
```

Related commands

`TRACe:CLEar`, `FETCh:ARRay?`, `FORMat[:DATA]`, `INITiate`

17

CHAPTER 17

TRIGger commands

This chapter describes the trigger model. There is exactly one trigger sequence for all channels: INITiate arms the triggered operation with its participants, the operation waits for one event of the source selected with TRIGger[:SEQuence]:SOURce, and the accepted event defines the common instant t0 on the 1 us system timebase at which acquisition and DAC playback start. Channel-bound events name the observed channel with a numeric suffix (VOLTage<n>, CC<n>); the level and slope of the voltage and current events, the absolute time target and the LXI event name are configured once. The external trigger input is the Digital-I/O line routed to TINPut, the trigger output the line routed to TOUTput. Trigger Out is not a participant: it emits one pulse per accepted trigger, which marks t0, and has no source, enable or arm state of its own; routing the line is the enable. The edge of the input and the polarity and width of the output pulse are configured in the DIGital chapter with DIGital:TINPut:SLOPe, DIGital:TOUTput:POLarity and DIGital:TOUTput:PWIDth. The delay from the selected input edge to t0 and the delay from the accepted trigger to the start of the output pulse are product specifications of the data sheet. INITiate, ABORt and *TRG are described in their own chapters, the position of the acquisition window relative to t0 is SENSe:SWEep:OFFSet:POINts.

IN THIS CHAPTER

17.1 TRIGger:CURRent:SLOPe 357

17.2 TRIGger:CURRent[:LEVel] 358

17.3 TRIGger:LXI:EVENt 359

17.4 TRIGger:TIME 361

17.5 TRIGger:VOLTage:SLOPe 362

17.6 TRIGger:VOLTage[:LEVel] 363

17.7 TRIGger[:SEQuence]:SOURce 365

17.1 TRIGger:CURRent:SLOPe

Selects or queries the crossing direction of the current level event.

The current event fires when the measured current of the channel named by the CURRent<n> suffix crosses the level of CURRent[:LEVel] in this direction: POSitive is a crossing from below the level to at or above it, NEGative a crossing from above to at or below it. The slope is configured once and applies whichever channel the suffix selects. A current already on the active side of the level when the operation is armed does not satisfy the event; a new crossing after arming is required. The detector may use a small fixed hysteresis; it is not programmable. The slope is frozen while an armed or active operation triggers on the current event.

COMMAND

```
TRIGger:CURRent:SLOPe
  POSitive|NEGative
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
POSitive	CPD		Crossing from below the level to at or above it.
NEGative	CPD		Crossing from above the level to at or below it.

QUERY

```
TRIGger:CURRent:SLOPe?
```

Returns

<CRD> POS or NEG.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation whose trigger source is CURRent<n> is armed or active. The slope is unchanged.

Example

```
TRIG:CURR:SLOP NEG
TRIG:CURR:SLOP?
-> NEG
```

Related commands

`TRIGger:CURRent[:LEVel]`, `TRIGger[:SEQuence]:SOURce`, `TRIGger:VOLTage:SLOPe`

17.2 TRIGger:CURRent[:LEVel]

Sets or queries the level of the current level event.

The level is a signed measured-current value in amperes: the event fires when the measured current of the channel named by the `CURRent<n>` suffix crosses it in the direction of `CURRent:SLOPe`. It is independent of the non-negative current limit of `[SOURce:]CURRent` and therefore works in sourcing and in sinking operation, where the measured current is negative. The level is configured once, without a channel list, and applies whichever channel the suffix selects. Programming the level neither arms nor triggers anything. It is frozen while an armed or active operation triggers on the current event; the query returns the programmed level, not a measurement.

COMMAND

```
TRIGger:CURRent[:LEVel]
  <current>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<current>`	NRf, A	Model-dependent	Signed measured current at which the event fires.
`MINimum`	CPD		Most negative programmable level.
`MAXimum`	CPD		Most positive programmable level.
`DEFault`	CPD		Factory default, 0 A.

QUERY

```
TRIGger:CURRent[:LEVel]?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the most negative programmable level instead of the setting.
`MAXimum`	CPD		Return the most positive programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 A, instead of the setting.

Returns

`<NR3>, A`	The programmed level, or the selected limit or default. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation whose trigger source is CURRent<n> is armed or active. The level is unchanged.
-222	"Data out of range"	<current> is outside the programmable range. The level is unchanged.

Example

```
TRIG:CURR 1.5
TRIG:CURR?
-> 1.500000000E+00
TRIG:CURR? DEF
-> 0.000000000E+00
```

Related commands

TRIGger:CURRent:SLOPe, TRIGger[:SEQuence]:SOURce, TRIGger:VOLTage[:LEVel], [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

17.3 TRIGger:LXI:EVENt

Sets or queries the name of the received LXI event that produces the LXI trigger event.

Only a received LXI event whose event ID matches this name and whose domain is the configured LXI event domain produces the local LXI event; any other received event is ignored for triggering. The name is a quoted ASCII string of 1 to 16 characters; the standard IDs "LAN0" to "LAN7" and application-specific names are accepted, names with the reserved LXI prefix remain subject to the LXI specification. A received event with a zero timestamp means now, a timestamped event may be scheduled against the same absolute clock that the time trigger uses. The name is frozen while an armed or active operation triggers on the LXI event. The transmit side of LXI event messaging is not part of this command set (TBD).

COMMAND

```
TRIGger:LXI:EVENt
  <event-id>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<event-id>	String		LXI event ID in quotes, 1 to 16 ASCII characters, for example "LAN0".

QUERY

```
TRIGger:LXI:EVENt?
```

Returns

<String> The configured event ID in quotes.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation whose trigger source is LXI is armed or active. The event ID is unchanged.
-224	"Illegal parameter value"	<event-id> is empty. The event ID is unchanged.
-150	"String data error"	<event-id> is longer than 16 characters. The event ID is unchanged.

Example

```
TRIG:LXI:EVEN "LAN2"
TRIG:LXI:EVEN?
-> "LAN2"
TRIG:SOUR LXI
INIT ACQ
```

Related commands

TRIGger[:SEQuence]:SOURce, SYSTem:COMMunicate:LXI:EVENt:DOMain, INITiate

17.4 TRIGger:TIME

Sets or queries the absolute UTC instant of the time trigger event.

The target is one absolute instant in UTC, independent of the local time zone, evaluated against the absolute system clock: the battery-backed clock, disciplined by PTP while synchronization is valid. When the clock reaches the target one TIME event is produced; the same target never produces a second event, and a new setting establishes a new target. A target already in the past produces no event and therefore no Trigger-Out pulse, and is accepted by this command without error: the check happens at INITiate, which requires a target strictly in the future and a valid clock. Loss of PTP synchronization after arming does not abort the operation, the clock continues from its last disciplined state and SYSTem:TIME:PTP:STATus? reports HOLDOVER; the 1 ns programming resolution of the seconds field is not a statement of trigger accuracy. The target is frozen while an armed or active operation triggers on the time event. A clock step that moves the absolute clock past the target, by SYSTem:TIME, SYSTem:DATE or PTP, produces the event at once, exactly once; a step back before the target does not produce it again.

COMMAND

```
TRIGger:TIME
  <year>,<month>,<day>,<hour>,<minute>,<seconds>
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
<year>	NR1		Year of the target in UTC, four digits.
<month>	NR1	1 to 12	Month of the target.
<day>	NR1	1 to 31	Day of the month.
<hour>	NR1	0 to 23	Hour of the target in UTC.
<minute>	NR1	0 to 59	Minute of the target.
<seconds>	NRf, s	0 to less than 60	Seconds of the target, with up to nine decimal places.

QUERY

```
TRIGger:TIME?
```

Returns

<NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR2>	The programmed target as <year>, <month>, <day>, <hour>, <minute>, <seconds> in UTC, the seconds with enough fractional digits to reproduce the programmed target.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation whose trigger source is TIME is armed or active. The target is unchanged.
-222	"Data out of range"	A field is outside its range, or the fields do not form a valid calendar date. The target is unchanged.

Example

```
TRIG:TIME 2026,8,17,13,30,12.5
TRIG:TIME?
-> 2026,8,17,13,30,12.500000000
TRIG:SOUR TIME
INIT ACQ
```

Related commands

TRIGger[:SEQuence]:SOURce, SYSTem:TIME, SYSTem:TIME:PTP:STATus?, INITiate

17.5 TRIGger:VOLTage:SLOPe

Selects or queries the crossing direction of the voltage level event.

The voltage event fires when the measured voltage of the channel named by the VOLTage<n> suffix crosses the level of VOLTage[:LEVel] in this direction: POSitive is a crossing from below the level to at or above it, NEGative a crossing from above to at or below it. The slope is configured once and applies whichever channel the suffix selects. A voltage already on the active side of the level when the operation is armed does not satisfy the event; a new crossing after arming is required. The detector may use a small fixed hysteresis; it is not programmable. The slope is frozen while an armed or active operation triggers on the voltage event.

COMMAND

```
TRIGger:VOLTage:SLOPe
  POSitive|NEGative
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
POSitive	CPD		Crossing from below the level to at or above it.
NEGative	CPD		Crossing from above the level to at or below it.

QUERY

```
TRIGger:VOLTage:SLOPe?
```

Returns

<CRD>	POS or NEG.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation whose trigger source is VOLTage<n> is armed or active. The slope is unchanged.

Example

```
TRIG:VOLT:SLOP NEG
TRIG:VOLT:SLOP?
-> NEG
```

Related commands

TRIGger:VOLTage[:LEVel], TRIGger[:SEQuence]:SOURce, TRIGger:CURRent:SLOPe

17.6 TRIGger:VOLTage[:LEVel]

Sets or queries the level of the voltage level event.

The level is a measured-voltage value in volts: the event fires when the measured voltage of the channel named by the VOLTage<n> suffix crosses it in the direction of VOLTage:SLOPe. It is independent of the voltage setpoint of [SOURce:]VOLTage. The level is configured once, without a channel list, and applies whichever channel the suffix selects. Programming the level neither arms nor triggers anything. It is frozen while an armed or active operation triggers on the voltage event; the query returns the programmed level, not a measurement.

COMMAND

```
TRIGger:VOLTage[:LEVel]
  <voltage>|MINimum|MAXimum|DEFault
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`<voltage>`	NRf, V	Model-dependent	Measured voltage at which the event fires.
`MINimum`	CPD		Lowest programmable level.
`MAXimum`	CPD		Highest programmable level.
`DEFault`	CPD		Factory default, 0 V.

QUERY

```
TRIGger:VOLTage[:LEVel]?
  [MINimum|MAXimum|DEFault]
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`MINimum`	CPD		Return the lowest programmable level instead of the setting.
`MAXimum`	CPD		Return the highest programmable level instead of the setting.
`DEFault`	CPD		Return the factory default, 0 V, instead of the setting.

Returns

`<NR3>, V` The programmed level, or the selected limit or default. Not a measurement.

Errors

CODE	MESSAGE	CONDITION
-221	`"Settings conflict"`	An operation whose trigger source is `VOLTage<n>` is armed or active. The level is unchanged.
-222	`"Data out of range"`	`<voltage>` is outside the programmable range. The level is unchanged.

Example

```
TRIG:VOLT 4.5
TRIG:VOLT?
-> 4.500000000E+00
TRIG:VOLT? DEF
-> 0.000000000E+00
```

Related commands

`TRIGger:VOLTage:SLOPe`, `TRIGger[:SEQuence]:SOURce`, `TRIGger:CURRent[:LEVel]`, `MEASure[:SCALar]:VOLTage[:DC]?`

17.7 TRIGger[:SEQuence]:SOURce

Selects or queries the event that starts the armed triggered operation.

The instrument has one trigger sequence, so the selection applies to every participant of the operation armed with `INITiate`: the acquisition into the record or into a file and the DAC playback all start at the same t0, defined by the first event of the selected source after arming; later events of the same source do not retrigger anything. Channel-bound sources name the observed channel with a numeric suffix, omitted meaning channel 1; the observed channel need not participate in the operation, and a suffix beyond the channel count of the model is an undefined header. Level and slope of the voltage and current events, the time target and the LXI event name are configured with the other commands of this chapter, the edge of the external trigger input with `DIGital:TINPut:SLOPe`; CV, CC and protection events have no settings and require a new transition or trip after arming. Whatever the source, the accepted trigger also produces the Trigger-Out pulse that marks t0 on the line routed to `TOUTput`. Selecting a source is always legal by itself; a conflict with the participants or with `SENSe:SWEep:OFFSet:POINts` is detected by `INITiate`, which generates -221 and arms nothing. The source is frozen while an operation is armed or active.

COMMAND

```
TRIGger[:SEQuence]:SOURce
  IMMediate|BUS|EXTernal|VOLTage<n>|CURRent<n>|CV<n>|CC<n>|OVP<n>|OCP<n>|OPP<n>|
  TIME|LXI
```

Parameters

NAME	TYPE / UNIT	RANGE	DESCRIPTION
`IMMediate`	CPD		The operation starts as soon as it is armed; t0 is the first slot of the 1 us timebase after a successful `INITiate`. Waiting for trigger is passed through.
`BUS`	CPD		Wait for `*TRG` or the equivalent VXI-11, USB TMC or HiSLIP trigger operation.
`EXTernal`	CPD		Wait for the edge selected with `DIGital:TINPut:SLOPe` on the Digital-I/O line routed to `TINPut`. Slope and routing are accepted also while the operation is armed or active; a new slope applies from the next edge, and without a routed line no edge comes, so the operation waits until `ABORt`.
`VOLTage<n>`	CPD		Wait for the measured voltage of channel <n> to cross `VOLTage[:LEVel]` in the direction of `VOLTage:SLOPe`. Omitted suffix: channel 1.

CURRent<n>	CPD	Wait for the measured current of channel <n> to cross CURRent[:LEVel] in the direction of CURRent:SLOPe. Omitted suffix: channel 1.
CV<n>	CPD	Wait for a new transition of channel <n> into constant-voltage regulation. Omitted suffix: channel 1.
CC<n>	CPD	Wait for a new transition of channel <n> into constant-current regulation. Omitted suffix: channel 1.
OVP<n>	CPD	Wait for a new overvoltage-protection trip of channel <n>. Operations without TRANsient only. Omitted suffix: channel 1.
OCP<n>	CPD	Wait for a new overcurrent-protection trip of channel <n>. Operations without TRANsient only. Omitted suffix: channel 1.
OPP<n>	CPD	Wait for a new overpower-protection trip of channel <n>. Operations without TRANsient only. Omitted suffix: channel 1.
TIME	CPD	Wait until the absolute clock reaches the UTC target of TRIGger:TIME; the target must be in the future at INITiate.
LXI	CPD	Wait for a received LXI event matching TRIGger:LXI:EVENt and the configured domain.

QUERY

```
TRIGger[:SEQuence]:SOURce?
```

Returns

<CRD>	IMM, BUS, EXT, VOLT<n>, CURR<n>, CV<n>, CC<n>, OVP<n>, OCP<n>, OPP<n>, TIME or LXI; channel-bound sources with their suffix, for example VOLT1.

Errors

CODE	MESSAGE	CONDITION
-221	"Settings conflict"	An operation is armed or active. The source is unchanged.

Example

```
TRIG:SOUR BUS
TRIG:SOUR?
-> BUS
TRIG:SOUR CURR2
TRIG:SOUR?
-> CURR2
```

Related commands

`INITiate`, `DIGital:TINPut:SLOPe`, `SENSe:SWEep:OFFSet:POINts`, `*TRG`

A

APPENDIX A

Reset and persistence

The instrument has three kinds of settings: volatile configuration that *RST returns to a defined state, saved setups that *SAV and *RCL store in ten slots, and persistent system settings that survive both. This appendix lists the reset value of every setting and whether it is part of a saved setup.

IN THIS CHAPTER

A.1 What *RST, *SAV, *RCL and power-up do 368

A.2 Reset values 369

A.1 What *RST, *SAV, *RCL and power-up do

***RST**	Puts all volatile configuration into the reset state listed below, switches every output off, aborts a running triggered operation and clears the OVP, OCP and OPP trip latches and the OTP, INHIBIT and HW_FAULT latches whose cause is gone; OTP and HW_FAULT stay set while their cause persists, and INHIBIT is cleared because the inhibit mode returns to OFF, after which the input no longer counts. It also clears the acquisition record and the stream cursor, empties the statistics window and cancels the pending *OPC and *OPC? requests of every session. It routes every Digital-I/O line back to GPIO and releases it; every function is stopped first, as by `DIGital:MODE`: an open I2C transaction ends with STOP, the counters read 0, the UART receive buffers and error flags are discarded and a PWM stops at its inactive level. After *RST no Digital-I/O function runs, and the inhibit input has no effect until `OUTPut:INHibit:MODE` is programmed again; a controller that uses it as an interlock programs routing, level and mode after every *RST. It does not touch the error queue, the status enable masks, saved setups or persistent system settings.
`*SAV <slot>`	Stores the current configuration atomically in slot 0 to 9, overwriting the previous contents. The Digital-I/O settings, the line routing included, are part of a setup. The output state, measurement data, statistics, status history, calibration and communication settings are never part of a setup.

`*RCL <slot>`	Aborts running operations, switches the outputs off, then restores the configuration of the slot. The Digital-I/O functions of the lines that lose them are stopped first, in the same order as by `DIGital:MODE`, then every other Digital-I/O value of the slot is applied, and the lines that receive a function are routed last; a recalled `PWM` or `TOUTput` routing is then active at once with the recalled values. Outputs stay off until enabled explicitly.
Power-up	Starts from the reset state for volatile configuration and restores the persistent system settings. Every Digital-I/O line starts released and routed to `GPIO`, and every Digital-I/O setting, the inhibit level and mode included, starts at its reset value; no Digital-I/O configuration survives a power cycle. No setup slot is recalled automatically.
Persistent settings	LAN, USB and RS-232 configuration, LXI event domain, clock and time zone, mains frequency, display, beeper and CSV preferences. Changed only by their own commands, unaffected by *RST, *SAV and *RCL.

A.2 Reset values

SETTING	*RST	SAVED BY *SAV
IEEE-488.2		
*ESE	Unchanged; power-up sets 0	No
*SRE	Unchanged; power-up sets 0	No
DIGITAL		
`DIGital:COUNter:SLOPe`	POSitive	Yes
`DIGital:FAULt:ACTive`	HIGH	Yes
`DIGital:FAULt:SOURce`	ALL	Yes
`DIGital:I2C:FREQuency`	100 kHz	Yes
`DIGital:LINE<n>:MODE`	GPIO on every line	Yes
`DIGital:PWM<i>:ACTive`	HIGH	Yes
`DIGital:PWM<i>:DCYCle`	50 %	Yes
`DIGital:PWM<i>:FREQuency`	1 kHz	Yes
`DIGital:SPI:CPHase`	0	Yes
`DIGital:SPI:CPOLarity`	0	Yes
`DIGital:SPI:CS<c>:ACTive`	LOW	Yes
`DIGital:SPI:FREQuency`	100 kHz	Yes
`DIGital:SPI:ORDer`	MSB	Yes
`DIGital:STATe`	63 (every line released)	Yes

DIGital:TINPut:SLOPe	NEGative	Yes
DIGital:TOUTput:POLarity	NEGative	Yes
DIGital:TOUTput:PWIDth	10 us	Yes
DIGital:UART:BAUD	9600 bit/s	Yes
DIGital:UART:BITS	8	Yes
DIGital:UART:PARity	NONE	Yes
DIGital:UART:SBITs	1	Yes
DISPLAY		
DISPlay:BRIGhtness	Unchanged, persistent	No
DISPlay:STATe	Unchanged, persistent	No
DISPlay[:WINDow]:TEXT[:DATA]	"" (message cleared)	No
FORMAT		
FORMat:BORDer	SWAPped	Yes
FORMat:SREGister	ASCii	Yes
FORMat[:DATA]	ASCii	Yes
MEASURE		
MEASure[:SCALar]:ENERgy:UNIT	WS	Yes
OUTPUT		
OUTPut:INHibit:ACTive	HIGH	Yes
OUTPut:INHibit:MODE	OFF	Yes
OUTPut:MODE	AUTO	Yes
OUTPut[:STATe]	OFF	No, always off after recall
SENSE		
SENSe:CURRent:RANGe	Range selected by autorange (RANGe:AUTO ON)	Yes
SENSe:CURRent:RANGe:AUTO	ON	Yes
SENSe:FUNCtion:CURRent	ON on every channel	Yes
SENSe:FUNCtion:VOLTage	ON on every channel	Yes
SENSe:SWEep:OFFSet:POINts	0	Yes
SENSe:SWEep:POINts	1	Yes
SENSe:SWEep:TINTerval	10 us	Yes
SENSe:VOLTage:RANGe	Range selected by autorange (RANGe:AUTO ON)	Yes
SENSe:VOLTage:RANGe:AUTO	ON	Yes
SOURCE		
[SOURce:]ARB:EXPonential:END:TIME	0 s	Yes
[SOURce:]ARB:EXPonential:END[:LEVel]	0 V	Yes

`[SOURce:]ARB:EXPonential:STARt:TIME`	0 s	Yes
`[SOURce:]ARB:EXPonential:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:EXPonential:TCONstant`	1 ms	Yes
`[SOURce:]ARB:EXPonential:TIME`	1 ms	Yes
`[SOURce:]ARB:FUNCtion:SHAPe`	PULSe	Yes
`[SOURce:]ARB:LOGStaircase:END:TIME`	0 s	Yes
`[SOURce:]ARB:LOGStaircase:END[:LEVel]`	0 V	Yes
`[SOURce:]ARB:LOGStaircase:NSTeps`	10	Yes
`[SOURce:]ARB:LOGStaircase:STARt:TIME`	0 s	Yes
`[SOURce:]ARB:LOGStaircase:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:LOGStaircase:TIME`	1 ms	Yes
`[SOURce:]ARB:MODE`	SEQuence	Yes
`[SOURce:]ARB:PULSe:HIGH`	0 V	Yes
`[SOURce:]ARB:PULSe:HIGH:TIME`	1 ms	Yes
`[SOURce:]ARB:PULSe:LOW`	0 V	Yes
`[SOURce:]ARB:PULSe:LOW:TIME`	1 ms	Yes
`[SOURce:]ARB:RAMP:END:TIME`	0 s	Yes
`[SOURce:]ARB:RAMP:END[:LEVel]`	0 V	Yes
`[SOURce:]ARB:RAMP:RTIMe`	1 ms	Yes
`[SOURce:]ARB:RAMP:STARt:TIME`	0 s	Yes
`[SOURce:]ARB:RAMP:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:SEQuence:COUNt`	1	Yes
`[SOURce:]ARB:SEQuence:DATA`	Empty list, `DATA:POINts?` returns 0	No, waveform data
`[SOURce:]ARB:SEQuence:FILE`	"" (no file)	Yes
`[SOURce:]ARB:SEQuence:PAUSe`	0 s	Yes
`[SOURce:]ARB:SEQuence:SOURce`	FILE	Yes
`[SOURce:]ARB:SINusoid:AMPLitude`	0 V	Yes
`[SOURce:]ARB:SINusoid:FREQuency`	1 kHz	Yes
`[SOURce:]ARB:SINusoid:OFFSet`	0 V	Yes
`[SOURce:]ARB:SINusoid:PHASe`	0 deg	Yes
`[SOURce:]ARB:STAircase:END:TIME`	0 s	Yes
`[SOURce:]ARB:STAircase:END[:LEVel]`	0 V	Yes
`[SOURce:]ARB:STAircase:NSTeps`	10	Yes
`[SOURce:]ARB:STAircase:STARt:TIME`	0 s	Yes
`[SOURce:]ARB:STAircase:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:STAircase:TIME`	1 ms	Yes
`[SOURce:]ARB:STReam:POINts`	0	Yes

`[SOURce:]ARB:TERMinate:LAST`	OFF	Yes
`[SOURce:]ARB:TINTerval`	DEFault (TBD)	Yes
`[SOURce:]ARB:TRAPezoid:END:TIME`	0 s	Yes
`[SOURce:]ARB:TRAPezoid:FTIMe`	1 ms	Yes
`[SOURce:]ARB:TRAPezoid:RTIMe`	1 ms	Yes
`[SOURce:]ARB:TRAPezoid:STARt:TIME`	0 s	Yes
`[SOURce:]ARB:TRAPezoid:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:TRAPezoid:TOP:TIME`	0 s	Yes
`[SOURce:]ARB:TRAPezoid:TOP[:LEVel]`	0 V	Yes
`[SOURce:]ARB:TRIangle:FTIMe`	1 ms	Yes
`[SOURce:]ARB:TRIangle:RTIMe`	1 ms	Yes
`[SOURce:]ARB:TRIangle:STARt[:LEVel]`	0 V	Yes
`[SOURce:]ARB:TRIangle:TOP[:LEVel]`	0 V	Yes
`[SOURce:]ARB[:STATe]`	ON on the single-channel model, OFF on every channel of a multi-channel model	Yes
`[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]`	0 A	Yes
`[SOURce:]CURRent:PROTection:DELay`	0.05 s	Yes
`[SOURce:]CURRent:PROTection:LEVel`	MAXimum of the model	Yes
`[SOURce:]CURRent:PROTection[:STATe]`	OFF	Yes
`[SOURce:]POWer:PROTection:DELay`	0.05 s	Yes
`[SOURce:]POWer:PROTection:LEVel`	MAXimum of the model	Yes
`[SOURce:]POWer:PROTection[:STATe]`	OFF	Yes
`[SOURce:]VOLTage:PROTection:LEVel`	MAXimum of the model	Yes
`[SOURce:]VOLTage:PROTection[:STATe]`	OFF	Yes
`[SOURce:]VOLTage:SENSe[:SOURce]`	INTernal	Yes
`[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]`	0 V	Yes
STATUS		
`STATus:MEASurement:ENABle`	Unchanged; `STATus:PRESet` and power-up set 0	No
`STATus:MEASurement:NTRansition`	Unchanged; `STATus:PRESet` and power-up set 0	No

`STATus:MEASurement:PTRansition`	Unchanged; `STATus:PRESet` and power-up set 65535	No
`STATus:OPERation:ENABle`	Unchanged; `STATus:PRESet` and power-up set 0	No
`STATus:OPERation:NTRansition`	Unchanged; `STATus:PRESet` and power-up set 0	No
`STATus:OPERation:PTRansition`	Unchanged; `STATus:PRESet` and power-up set 65535	No
`STATus:QUEStionable:ENABle`	Unchanged; `STATus:PRESet` and power-up set 0	No
`STATus:QUEStionable:NTRansition`	Unchanged; `STATus:PRESet` and power-up set 0	No
`STATus:QUEStionable:PTRansition`	Unchanged; `STATus:PRESet` and power-up set 65535	No
`STATus:QUEue:DISable`	Unchanged; power-up discards no code	No
`STATus:QUEue:ENABle`	Unchanged; power-up enables every code	No
SYSTEM		
`SYSTem:BEEPer:STATe`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:DESCription`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:HOSTname`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV4:ADDRess`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV4:GATeway`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV4:MODE`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV4:SUBNetmask`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV6:ADDRess`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV6:GATeway`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV6:MODE`	Unchanged, persistent	No
`SYSTem:COMMunicate:LAN:IPV6:PREFix`	Unchanged, persistent	No
`SYSTem:COMMunicate:LXI:EVENt:DOMain`	Unchanged, persistent	No
`SYSTem:COMMunicate:RLSTate`	Unchanged; power-up sets LOC	No
`SYSTem:COMMunicate:SERial:BAUDrate`	Unchanged, persistent	No

SYSTem:COMMunicate:SERial:FLOWcontrol	Unchanged, persistent	No
SYSTem:COMMunicate:USB:CLASs	Unchanged, persistent	No
SYSTem:CSV:DECimal	Unchanged, persistent	No
SYSTem:CSV:DELimiter	Unchanged, persistent	No
SYSTem:CSV:LINeending	Unchanged, persistent	No
SYSTem:DATE	Unchanged, persistent	No
SYSTem:LFRequency	Unchanged, persistent	No
SYSTem:TIME	Unchanged, persistent	No
SYSTem:TZONe	Unchanged, persistent	No
TRIGGER		
TRIGger:CURRent:SLOPe	POSitive	Yes
TRIGger:CURRent[:LEVel]	DEFault (0 A)	Yes
TRIGger:LXI:EVENt	"LAN0"	Yes
TRIGger:TIME	1970,1,1,0,0,0.000000000	Yes
TRIGger:VOLTage:SLOPe	POSitive	Yes
TRIGger:VOLTage[:LEVel]	DEFault (0 V)	Yes
TRIGger[:SEQuence]:SOURce	IMMediate	Yes

Index

*CLS, 10
*ESE, 10
*ESR?, 11
*IDN?, 12
*OPC, 13
*OPT?, 14
*RCL, 15
*RST, 16
*SAV, 16
*SRE, 17
*STB?, 18
*TRG, 19
*TST?, 20
*WAI, 21
[SOURce:]ARB:EXPonential:END:TIME, 168
[SOURce:]ARB:EXPonential:END[:LEVel], 169
[SOURce:]ARB:EXPonential:STARt:TIME, 171
[SOURce:]ARB:EXPonential:STARt[:LEVel], 173
[SOURce:]ARB:EXPonential:TCONstant, 174
[SOURce:]ARB:EXPonential:TIME, 176
[SOURce:]ARB:FUNCtion:SHAPe, 178
[SOURce:]ARB:LOGStaircase:END:TIME, 179
[SOURce:]ARB:LOGStaircase:END[:LEVel], 181
[SOURce:]ARB:LOGStaircase:NSTeps, 183
[SOURce:]ARB:LOGStaircase:STARt:TIME, 184
[SOURce:]ARB:LOGStaircase:STARt[:LEVel], 186
[SOURce:]ARB:LOGStaircase:TIME, 188
[SOURce:]ARB:MODE, 189
[SOURce:]ARB:PULSe:HIGH, 190
[SOURce:]ARB:PULSe:HIGH:TIME, 192
[SOURce:]ARB:PULSe:LOW, 194
[SOURce:]ARB:PULSe:LOW:TIME, 195
[SOURce:]ARB:RAMP:END:TIME, 197
[SOURce:]ARB:RAMP:END[:LEVel], 199
[SOURce:]ARB:RAMP:RTIMe, 200
[SOURce:]ARB:RAMP:STARt:TIME, 202
[SOURce:]ARB:RAMP:STARt[:LEVel], 204
[SOURce:]ARB:SEQuence:COUNt, 205
[SOURce:]ARB:SEQuence:DATA, 207
[SOURce:]ARB:SEQuence:DATA:POINts?, 209
[SOURce:]ARB:SEQuence:FILE, 210
[SOURce:]ARB:SEQuence:PAUSe, 212
[SOURce:]ARB:SEQuence:SOURce, 213
[SOURce:]ARB:SINusoid:AMPLitude, 215
[SOURce:]ARB:SINusoid:FREQuency, 217
[SOURce:]ARB:SINusoid:OFFSet, 218
[SOURce:]ARB:SINusoid:PHASe, 220

[SOURce:]ARB:STAircase:END:TIME, 222
[SOURce:]ARB:STAircase:END[:LEVel], 223
[SOURce:]ARB:STAircase:NSTeps, 225
[SOURce:]ARB:STAircase:STARt:TIME, 227
[SOURce:]ARB:STAircase:STARt[:LEVel], 228
[SOURce:]ARB:STAircase:TIME, 230
[SOURce:]ARB:STReam:DATA, 232
[SOURce:]ARB:STReam:DATA:CURRent, 233
[SOURce:]ARB:STReam:DATA:VOLTage, 234
[SOURce:]ARB:STReam:POINts, 236
[SOURce:]ARB:TERMinate:LAST, 237
[SOURce:]ARB:TINTerval, 238
[SOURce:]ARB:TRAPezoid:END:TIME, 240
[SOURce:]ARB:TRAPezoid:FTIMe, 241
[SOURce:]ARB:TRAPezoid:RTIMe, 243
[SOURce:]ARB:TRAPezoid:STARt:TIME, 245
[SOURce:]ARB:TRAPezoid:STARt[:LEVel], 246
[SOURce:]ARB:TRAPezoid:TOP:TIME, 248
[SOURce:]ARB:TRAPezoid:TOP[:LEVel], 249
[SOURce:]ARB:TRIangle:FTIMe, 251
[SOURce:]ARB:TRIangle:RTIMe, 253
[SOURce:]ARB:TRIangle:STARt[:LEVel], 254
[SOURce:]ARB:TRIangle:TOP[:LEVel], 256
[SOURce:]ARB[:STATe], 258
[SOURce:]CURRent:PROTection:CLEar, 261
[SOURce:]CURRent:PROTection:DELay, 262
[SOURce:]CURRent:PROTection:LEVel, 264
[SOURce:]CURRent:PROTection:TRIPped?, 265
[SOURce:]CURRent:PROTection[:STATe], 266
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude], 259
[SOURce:]POWer:PROTection:CLEar, 268
[SOURce:]POWer:PROTection:DELay, 269
[SOURce:]POWer:PROTection:LEVel, 270
[SOURce:]POWer:PROTection:TRIPped?, 272
[SOURce:]POWer:PROTection[:STATe], 273
[SOURce:]VOLTage:PROTection:CLEar, 274
[SOURce:]VOLTage:PROTection:LEVel, 275
[SOURce:]VOLTage:PROTection:TRIPped?, 277
[SOURce:]VOLTage:PROTection[:STATe], 278
[SOURce:]VOLTage:SENSe[:SOURce], 280
[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude], 281

ABORt, 22

DIGital:COUNter:CLEar, 29
DIGital:COUNter:DATA?, 30
DIGital:COUNter:SLOPe, 30
DIGital:FAULt:ACTive, 31
DIGital:FAULt:SOURce, 32
DIGital:I2C:FREQuency, 34
DIGital:I2C:READ?, 35
DIGital:I2C:WRITe, 37
DIGital:LINE<n>:MODE, 39
DIGital:LINE<n>:READ?, 40
DIGital:LINE<n>:STATe, 41
DIGital:MODE, 42

DIGital:PWM<i>:ACTive, 44
DIGital:PWM<i>:DCYCle, 45
DIGital:PWM<i>:FREQuency, 46
DIGital:READ?, 48
DIGital:SPI:CPHase, 48
DIGital:SPI:CPOLarity, 49
DIGital:SPI:CS<c>:ACTive, 50
DIGital:SPI:FREQuency, 51
DIGital:SPI:ORDer, 53
DIGital:SPI:TRANsfer?, 54
DIGital:SPI:WRITe, 55
DIGital:STATe, 57
DIGital:TINPut:SLOPe, 58
DIGital:TOUTput:POLarity, 59
DIGital:TOUTput:PWIDth, 60
DIGital:UART:AVAilable?, 62
DIGital:UART:BAUD, 62
DIGital:UART:BITS, 64
DIGital:UART:CLEar, 65
DIGital:UART:ERRor:CLEar, 66
DIGital:UART:ERRor?, 66
DIGital:UART:PARity, 67
DIGital:UART:READ?, 68
DIGital:UART:SBITs, 69
DIGital:UART:WRITe, 70
DISPlay:BRIGhtness, 72
DISPlay:STATe, 74
DISPlay[:WINDow]:TEXT:CLEar, 75
DISPlay[:WINDow]:TEXT[:DATA], 75

FETCh:ARRay:NEXT?, 79
FETCh:ARRay:POINts?, 80
FETCh:ARRay?, 77
FETCh[:SCALar]:CURRent[:DC]?, 81
FETCh[:SCALar]:POWer?, 82
FETCh[:SCALar]:VOLTage[:DC]?, 83
FORMat:BORDer, 85
FORMat:SREGister, 86
FORMat[:DATA], 88

HCOPy:DATA?, 90

INITiate, 23

MEASure:HISTogram:CURRent:BIN?, 93
MEASure:HISTogram:CURRent:DATA?, 94
MEASure:HISTogram:CURRent:EDGE?, 96
MEASure:HISTogram:CURRent:POINts?, 97
MEASure:HISTogram:POWer:BIN?, 98
MEASure:HISTogram:POWer:DATA?, 99
MEASure:HISTogram:POWer:EDGE?, 100
MEASure:HISTogram:POWer:POINts?, 101
MEASure:HISTogram:VOLTage:BIN?, 102
MEASure:HISTogram:VOLTage:DATA?, 103
MEASure:HISTogram:VOLTage:EDGE?, 105
MEASure:HISTogram:VOLTage:POINts?, 106

MEASure[:SCALar]:CURRent[:DC]:AVERage?, 108
MEASure[:SCALar]:CURRent[:DC]:MAXimum?, 109
MEASure[:SCALar]:CURRent[:DC]:MINimum?, 110
MEASure[:SCALar]:CURRent[:DC]:PTPeak?, 111
MEASure[:SCALar]:CURRent[:DC]?, 107
MEASure[:SCALar]:ENERgy:RESet, 113
MEASure[:SCALar]:ENERgy:UNIT, 114
MEASure[:SCALar]:ENERgy?, 112
MEASure[:SCALar]:POWer:AVERage?, 117
MEASure[:SCALar]:POWer:MAXimum?, 118
MEASure[:SCALar]:POWer:MINimum?, 119
MEASure[:SCALar]:POWer:PTPeak?, 120
MEASure[:SCALar]:POWer?, 115
MEASure[:SCALar]:STATistic:COUNt?, 121
MEASure[:SCALar]:STATistic:DURation?, 122
MEASure[:SCALar]:STATistic:RESet, 123
MEASure[:SCALar]:VOLTage[:DC]:AVERage?, 125
MEASure[:SCALar]:VOLTage[:DC]:MAXimum?, 126
MEASure[:SCALar]:VOLTage[:DC]:MINimum?, 127
MEASure[:SCALar]:VOLTage[:DC]:PTPeak?, 128
MEASure[:SCALar]:VOLTage[:DC]?, 124
MMEMory:CAPacity?, 130
MMEMory:DATA:APPend, 132
MMEMory:DATA[:DEFinite], 133
MMEMory:DELete, 135
MMEMory:FREE?, 136
MMEMory:INITialize, 138
MMEMory:LIST?, 139

OUTPut:INHibit:ACTive, 141
OUTPut:INHibit:MODE, 142
OUTPut:MODE, 144
OUTPut:PROTection:CLEar, 145
OUTPut[:STATe], 146

SENSe:CURRent:RANGe, 150
SENSe:CURRent:RANGe:AUTO, 152
SENSe:FUNCtion:CURRent, 153
SENSe:FUNCtion:VOLTage, 155
SENSe:SWEep:OFFSet:POINts, 156
SENSe:SWEep:POINts, 158
SENSe:SWEep:TINTerval, 159
SENSe:VOLTage:RANGe, 161
SENSe:VOLTage:RANGe:AUTO, 163
STATus:MEASurement:CONDition?, 295
STATus:MEASurement:ENABle, 295
STATus:MEASurement:NTRansition, 297
STATus:MEASurement:PTRansition, 298
STATus:MEASurement[:EVENt]?, 299
STATus:OPERation:CONDition?, 285
STATus:OPERation:ENABle, 285
STATus:OPERation:NTRansition, 287
STATus:OPERation:PTRansition, 288
STATus:OPERation[:EVENt]?, 289
STATus:PRESet, 300
STATus:QUEStionable:CONDition?, 290

STATus:QUEStionable:ENABle, 291
STATus:QUEStionable:NTRansition, 292
STATus:QUEStionable:PTRansition, 293
STATus:QUEStionable[:EVENt]?, 294
STATus:QUEue:CLEar, 300
STATus:QUEue:DISable, 301
STATus:QUEue:ENABle, 302
STATus:QUEue[:NEXT]?, 303
SYSTem:BEEPer:STATe, 307
SYSTem:BEEPer[:IMMediate], 308
SYSTem:COMMunicate:LAN:APPLy, 309
SYSTem:COMMunicate:LAN:DESCription, 309
SYSTem:COMMunicate:LAN:DESCription:STATus?, 311
SYSTem:COMMunicate:LAN:DISCard, 311
SYSTem:COMMunicate:LAN:HOSTname, 312
SYSTem:COMMunicate:LAN:HOSTname:STATus?, 313
SYSTem:COMMunicate:LAN:IPV4:ADDRess, 314
SYSTem:COMMunicate:LAN:IPV4:GATeway, 315
SYSTem:COMMunicate:LAN:IPV4:MODE, 316
SYSTem:COMMunicate:LAN:IPV4:STATus:ADDRess?, 318
SYSTem:COMMunicate:LAN:IPV4:STATus:GATeway?, 319
SYSTem:COMMunicate:LAN:IPV4:STATus:SUBNetmask?, 319
SYSTem:COMMunicate:LAN:IPV4:STATus?, 317
SYSTem:COMMunicate:LAN:IPV4:SUBNetmask, 320
SYSTem:COMMunicate:LAN:IPV6:ADDRess, 321
SYSTem:COMMunicate:LAN:IPV6:GATeway, 322
SYSTem:COMMunicate:LAN:IPV6:MODE, 323
SYSTem:COMMunicate:LAN:IPV6:PREFix, 324
SYSTem:COMMunicate:LAN:IPV6:STATus:GLOBal?, 326
SYSTem:COMMunicate:LAN:IPV6:STATus:LINKlocal?, 327
SYSTem:COMMunicate:LAN:IPV6:STATus?, 325
SYSTem:COMMunicate:LAN:MACaddress?, 328
SYSTem:COMMunicate:LAN:RESet, 328
SYSTem:COMMunicate:LXI:EVENt:DOMain, 329
SYSTem:COMMunicate:RLSTate, 330
SYSTem:COMMunicate:SELF?, 331
SYSTem:COMMunicate:SERial:BAUDrate, 332
SYSTem:COMMunicate:SERial:FLOWcontrol, 333
SYSTem:COMMunicate:USB:CLASs, 334
SYSTem:CSV:DECimal, 335
SYSTem:CSV:DELimiter, 336
SYSTem:CSV:LINeending, 337
SYSTem:DATE, 338
SYSTem:ERRor:ALL?, 339
SYSTem:ERRor:CLEar, 340
SYSTem:ERRor:CODE:ALL?, 341
SYSTem:ERRor:CODE[:NEXT]?, 341
SYSTem:ERRor:COUNt?, 342
SYSTem:ERRor[:NEXT]?, 343
SYSTem:LFRequency, 344
SYSTem:LOCal, 345
SYSTem:LOCK:OWNer?, 346
SYSTem:REMote, 346
SYSTem:RWLock, 347
SYSTem:TIME, 348

SYSTem:TIME:PTP:STATus?, 349
SYSTem:TIME:SOURce?, 350
SYSTem:TZONe, 350
SYSTem:VERSion?, 352

TRACe:CLEar, 353
TRACe:SAVE, 354
TRIGger:CURRent:SLOPe, 357
TRIGger:CURRent[:LEVel], 358
TRIGger:LXI:EVENt, 359
TRIGger:TIME, 361
TRIGger:VOLTage:SLOPe, 362
TRIGger:VOLTage[:LEVel], 363
TRIGger[:SEQuence]:SOURce, 365