Contents

About this Manual

This manual describes how to use the Standard Commands for Programming Instruments (SCPI) to communicate with the DAQ3120 Series Benchtop Data Acquisition System. It provides a comprehensive reference for the command syntax, parameters, and functioning of the instrument's remote control capabilities.

1.1 Intended Audience

This document is designed for instrument programmers and engineers tasked with creating SCPI-based software applications for the DAQ3120 Series. Users should have a basic understanding of remote communication interfaces (such as USB, LAN, or GPIB) and the SCPI programming language.

1.2 Related Documents

For additional information regarding the operation and standards of the instrument, refer to the following documents:

- **DAQ3120 Series User's Manual**: Describes the front-panel operation, installation, and general features of the DAQ3120 Series.
- **Standard Commands for Programming Instruments (SCPI)**: Volume 1-4, Version 1990.0 (May 1999), SCPI Consortium.
- **IEEE Std 488.2-1992**: IEEE Standard Codes, Formats, Protocols, and Common Commands. The Institute of Electrical and Electronics Engineers, Inc..

Syntax Convention

2.1 Introduction

SCPI (Standard Commands for Programmable Instruments) serves as a universal programming language designed for electronic test and measurement instruments. It is grounded in the IEEE 488.1 and IEEE 488.2 standards. The DAQ3120 series aligns with the SCPI language and incorporates the IEEE 488.2 STD status structure.
The commands can be issued over USB, LAN, or GPIB interfaces.

2.1.1 Types of SCPI Messages

In order to program a DAQ3120 instrument, it is necessary to create a program message. This message comprises one or more appropriately formatted SCPI commands transmitted from the controller to the DAQ3120 instrument.
The program message, which can be sent at any time, requests the instrument to execute a specific action or provide data or status information. These requests are also referred to as *queries*.
Upon receiving a query, the DAQ3120 instrument responds by sending a response message back to the controller. This response message contains data formatted in a specific SCPI format.

2.2 Types of SCPI Commands

Two types of SCPI commands are available: common commands, described below, and device-specific subsystem commands.

2.2.1 Common Commands

Common SCPI commands, as defined by IEEE 488.2, are responsible for controlling and managing generic system functions like reset, self-test, configuration storage, and device identification. Typically, common commands start with an asterisk (*), have a length of four to five characters, and may involve one or more parameters.
The command keyword is separated from the initial parameter by a space. Multiple commands can be separated using a semicolon (;), as demonstrated below:

```
*RST; *CLS; *ESE 32; *OPC?
```

Refer to Table 2.1 for a summary of the common SCPI commands applicable to the DAQ3120 series.

Command	Description
*CLS	Clears all Event Registers summarized in the status byte.
*ESE	Sets the Standard Event Status Enable Register bits.
*ESE?	Queries the Standard Event Status Enable Register.
*ESR?	Queries and clears the Standard Event Status Register.
*IDN?	Returns the unique identification string of the instrument.
*OPC	Sets the Operation Complete bit in the Standard Event Status Register when all pending operations are finished.
*OPC?	Returns a "1" to the output buffer when all pending operations are finished.
*PSC	Sets the Power-On Status Clear flag.
*PSC?	Queries the Power-On Status Clear flag.
*RCL	Restores the instrument settings from a specified memory location.
*RST	Resets the instrument to its default state.
*SAV	Saves the current instrument settings to a specified memory location.
*SRE	Sets the Service Request Enable Register bits.
*SRE?	Queries the Service Request Enable Register.
*STB?	Queries the Status Byte Register.
*TRG	Triggers the instrument if the trigger source is set to BUS.
*TST?	Performs a self-test and returns the result.
*WAI	Prevents the instrument from executing further commands until all pending operations are complete.

Table 2.1 Common SCPI Commands

2.3 Syntax of Program Messages

A program message consists of one or more properly formatted SCPI commands, a parameter (if necessary), and a terminator sent from the controller to the DAQ3120 instrument.

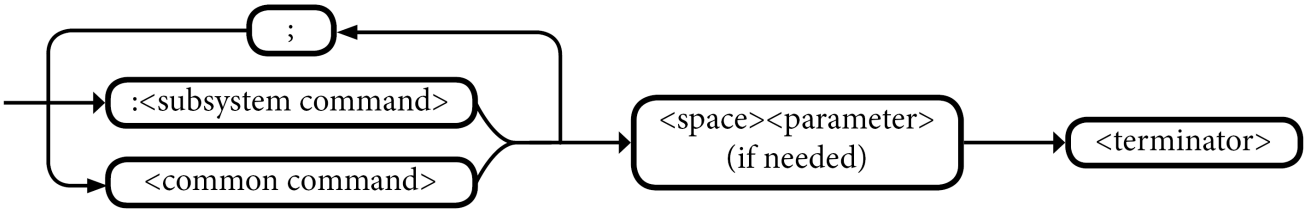

Figure 2.1 Syntax of Program Messages

2.3.1 Command Structure

SCPI commands follow a tree-like structure, organized into nodes. Each level of the command tree is a node. Each keyword (node) of a SCPI command is separated by a colon (:).

`CONFigure:VOLTage:DC`

2.3.2 Command Forms

Commands and queries have two different forms: long and short.

- **Long Form**: The full word (e.g., `CONFigure`).
- **Short Form**: An abbreviated version, typically the first three or four letters (e.g., `CONF`).

The commands can be written in either uppercase or lowercase. The instrument is case-insensitive. For example, `CONFIGURE`, `configure`, and `Conf` are all valid.

2.3.3 Square Brackets

Commands that contain square brackets `[]` indicate that the enclosed keywords or parameters are optional. The function of the command is the same with or without the bracketed items.

`[SENSe:]VOLTage:DC:RANGe`

Both `SENSe:VOLTage:DC:RANGe` and `VOLTage:DC:RANGe` are valid.

2.3.4 Separators and Terminators

- **Command Separator**: A semicolon (;) is used to link multiple commands within a single program message string.
- **Parameter Separator**: A space is used to separate the command header from the first parameter. A comma (,) is used to separate multiple parameters.
- **Message Terminator**: The termination character marks the end of a command line. The DAQ3120 series accepts the Line Feed (LF) character (<\n>) as the terminator. The IEEE 488.2 standard also defines CR, CR+LF, and LF+CR as valid terminators.

2.3.5 Automatic Parameter Range Selection

The instrument may automatically set a command parameter to the next available value if the entered value is not exact. For example, if a specific range is requested that does not exist, the instrument selects the next highest available range.

SCPI Data Types

SCPI defines various data types for use in program messages and response messages. The DAQ3120 Series uses a subset of these standard data types. This section summarizes the data types accepted by the instrument.

3.1 Data Types

The following data types are used in SCPI commands and queries.

3.1.1 Character Data Types

Character data types are short, mnemonic keywords used to represent specific settings.

- They can be entered in either long or short form (e.g., `IMMediate` or `IMM`).
- They are case-insensitive.
- Response messages always return the short form in uppercase.

```
Example: DISPlay:PAGE {MEAS | MANUMEM | INTF}
```

3.1.2 Numeric Data Types (<NRf>)

SCPI defines three formats for numeric values, collectively referred to as `<NRf>`. The instrument accepts any of these formats for numeric parameters.

Data	Description
<NR1>	Integer values (e.g., 0, 100, -10).
<NR2>	Decimal values (e.g., 1.5, 10.0, 0.01).
<NR3>	Floating point values with exponents (e.g., 1.5E+3, 4.5E-6).
<NRf>	Any of the above formats (Flexible Numeric Representation).

Table 3.1 Numeric Data Formats

3.1.3 Boolean Data Type (<Boolean>)

Boolean parameters represent a single binary condition (True or False).

- **ON** or **1**: Represents True (Enabled).
- **OFF** or **0**: Represents False (Disabled).

```
Example: OUTPut:STATe ON
```

3.1.4 String Data Type (<String>)

String parameters are enclosed in double quotes (e.g., "Label Name").

- Used for labels, messages, or filenames.
- The quotes are part of the parameter syntax.

```
Example: DISPlay:TEXT "Testing"
```

3.2 Special Parameter Keywords

In addition to standard values, SCPI commands often accept special keywords to set parameters to specific relative limits.

Parameter	Description
MIN	Sets the parameter to the minimum valid value for the current configuration.
MAX	Sets the parameter to the maximum valid value for the current configuration.
DEF	Sets the parameter to its default value.
AUTO	Enables automatic selection (e.g., Auto Range).

Table 3.2 Special Parameter Keywords

```
Example: VOLTage:DC:RANGe MAX
```

3.3 Channel List Parameter (@<ch_list>)

Many commands require specifying one or more channels. The channel list is enclosed in parentheses and preceded by an `@` symbol.

- **Single Channel**: `(@101)`
- **Multiple Channels**: `(@101, 102, 105)`
- **Range of Channels**: `(@101:105)` (Selects channels 101 through 105)
- **Combined**: `(@101:105, 201)`

About Commands & Queries

This section lists and describes the remote control commands and queries recognized by the instrument. All commands and queries can be executed in either local or remote state.

The description, command syntax, query syntax, example, and response are provided for each command. The commands are given in both long and short forms. All examples are shown in the short form. Queries perform actions such as obtaining information and are recognized by the question mark (?) following the header.

4.1 How They are Listed

The commands are listed by subsystem and in alphabetical order according to their short form.

4.2 How They are Described

In the descriptions themselves, a brief explanation of the function performed is given. This is followed by a presentation of the formal syntax, with the header given in Upper-and-Lower-Case characters and the short form derived from it in ALL UPPER-CASE characters. Where applicable, the syntax of the query is given with the format of its response.

4.3 When can they be used?

The commands and queries listed here can be used for the DAQ3120 Series.

4.4 Command Notation

The following notation is used in the command descriptions:

Notation	Description
< >	Angular brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command.
:=	A colon followed by an equals sign separates a placeholder from the description of the type and range of values that may be used in a command instead of the placeholder.
{ }	Braces enclose a list of choices, one of which must be made.
|	A vertical bar separates multiple parameter choices (e.g., {ON | OFF}).
[ ]	Square brackets enclose optional items.
...	An ellipsis indicates that the items both to its left and right may be repeated a number of times.

Table 4.1 Command Notation Symbols

Common Commands

Common commands are device-independent commands defined by the IEEE 488.2 standard. These commands control generic instrument functions such as reset, self-test, and status reporting. They always begin with an asterisk (*).

5.1 *CLS

Description: Clears the Status Byte register, all Event registers, and the Error Queue. This command does not clear the Enable registers or the output buffer.

Syntax:
Command *CLS

Parameters: None

Example:
Command *CLS

5.2 *ESE

Description: Sets or queries the bits in the Standard Event Status Enable Register. The enable register allows specific events in the Event Register to generate a summary bit in the Status Byte.

Syntax:
Command *ESE <NR1>
Query *ESE?

Parameters: <NR1> : 0 to 255

Example:
Command *ESE 32
Query *ESE?

5.3 *ESR?

Description: Queries and clears the Standard Event Status Register. The register is cleared after it is read.

Syntax:
Query *ESR?

Parameters: None

Example:
Query *ESR?

5.4 *IDN?

Description: Queries the instrument identification string.

Syntax:
Query `*IDN?`
Parameters: None
Example:
Query `*IDN?`

NOTICE **Returns: B&K Precision, DAQ3120, <Serial Number>, <Firmware Version>**

5.5 *OPC

Description: Sets the Operation Complete (OPC) bit (bit 0) in the Standard Event Status Register after all pending operations are completed.

Syntax:
Command `*OPC`
Query `*OPC?`
Parameters: None
Example:
Command `*OPC`
Query `*OPC?`

5.6 *PSC

Description: Sets the Power-On Status Clear flag. This controls whether the Enable Registers (Service Request Enable and Standard Event Status Enable) are cleared at power-on.

Syntax:
Command `*PSC <Boolean>`
Query `*PSC?`
Parameters: `<Boolean>` : 0 | 1
Example:
Command `*PSC 1`
Query `*PSC?`

5.7 *RCL

Description: Recalls the instrument state from a specified memory location.

Syntax:

Command *RCL <NR1>

Parameters: <NR1> : 0 to 5

Example:

Command *RCL 1

5.8 *RST

Description: Resets the instrument to its factory default state. This command aborts any scan in progress and clears the scan list.

Syntax:

Command *RST

Parameters: None

Example:

Command *RST

5.9 *SAV

Description: Saves the current instrument state to a specified memory location.

Syntax:

Command *SAV <NR1>

Parameters: <NR1> : 1 to 5

Example:

Command *SAV 2

5.10 *SRE

Description: Sets or queries the Service Request Enable Register. This determines which bits of the Status Byte can generate a Service Request (SRQ).

Syntax:
Command *SRE <NR1>
Query *SRE?
Parameters: <NR1> : 0 to 255
Example:
Command *SRE 16
Query *SRE?

5.11 *STB?

Description: Queries the Status Byte Register. The register is not cleared when read by this command.

Syntax:
Query *STB?
Parameters: None
Example:
Query *STB?

5.12 *TRG

Description: Triggers the instrument if the trigger source is set to BUS. Behavers similarly to the IEEE 488.1 Group Execute Trigger (GET).

Syntax:
Command *TRG
Parameters: None
Example:
Command *TRG

5.13 *TST?

Description: Performs a complete self-test of the instrument and returns the result.

Syntax:

Query *TST?

Parameters: None

Example:

Query *TST?

NOTICE **Returns 0 for pass or non-zero error code for fail.**

5.14 *WAI

Description: Configures the instrument's output buffer to wait until all pending operations are completed before executing any subsequent commands.

Syntax:

Command *WAI

Parameters: None

Example:

Command *WAI

Other Commands

This chapter describes the auxiliary commands used for controlling measurement acquisition, internal DMM state, and system units.

6.1 ABORt

Description: Aborts a measurement in progress from a scan and returns the instrument to the trigger idle state. If a scan is in progress, it stops immediately. Initiating a new scan after an abort will clear reading memory.

Syntax:

Command `ABORt`

Parameters: None

Example:

Command `ABORt`

6.2 FETCh?

Description: Waits for measurements to complete and copies all available measurements to the instrument's output buffer. Unlike `READ?`, this command does not erase readings from the memory, allowing multiple retrievals of the same data.

Syntax:

Query `FETCh?`

Parameters: None

Example:

Query `FETC?`

NOTICE

Reading memory can store up to 100,000 readings. If memory overflows, the oldest readings are overwritten. The `FETCh?` query returns data in the format specified by `FORMat:READing` commands.

6.3 INITiate[:IMMediate]

Description: Changes the state of the triggering system from "idle" to "wait-for-trigger" and clears the previous set of measurements from reading memory. Measurements begin when the specified trigger conditions are satisfied.

Syntax:

Command `INITiate[:IMMediate]`

Parameters: None

Example:

Command `INIT`

6.4 INSTrument:DMM

Description: Enables or disables the internal DMM. When disabled, the instrument acts as a switch, allowing external instruments to measure signals routed through the multiplexer modules.

Syntax:

Command `INSTrument:DMM <Boolean>`

Query `INSTrument:DMM?`

Parameters: `<Boolean>` : 0 | 1 | OFF | ON

Example:

Command `INST:DMM OFF`

Query `INST:DMM?`

NOTICE **Changing the state of the internal DMM triggers a Factory Reset (*RST).**

6.5 R?

Description: Reads and erases measurements from the reading memory up to the specified count, starting with the oldest measurement. This command is useful for preventing memory overflow during long scans by periodically removing read data.

Syntax:
Query `R? <NR1>`
Parameters: `<NR1>` : 1 to 100,000
Example:
Query `R? 10`

NOTICE **The query returns a block data string (e.g., #279...). Unlike `READ?` or `FETCh?`, `R?` does not wait for measurements to complete; it returns what is currently available.**

6.6 READ?

Description: Starts a new measurement scan and waits for it to complete before sending readings to the output buffer. This command is equivalent to sending `ABORt`, `INITiate`, and `FETCh?` in sequence.

Syntax:
Query `READ?`
Parameters: None
Example:
Query `READ?`

6.7 TIME:SYNC:SERVer

Description: Sets or queries the URL or IP address of the NTP (Network Time Protocol) server used to synchronize the system time.

Syntax:
Command `TIME:SYNC:SERVer <string>`
Query `TIME:SYNC:SERVer?`
Parameters: `<string>` : "URL" or "IP Address"
Example:
Command `TIME:SYNC:SERV "time.nist.gov"`
Query `TIME:SYNC:SERV?`

6.8 UNIT:TEMPerature

Description: Sets or queries the units for temperature measurements. This setting applies to the specified channels or all channels if no list is provided.

Syntax:

Command `UNIT:TEMPerature <units> [,<@ch_list>]`

Query `UNIT:TEMPerature? [<@ch_list>]`

Parameters: `<units>` : C | F | K `<@ch_list>` : (@101:105)

Example:

Command `UNIT:TEMP F,(@101)`

Query `UNIT:TEMP? (@101)`

Calculate Subsystem

The Calculate subsystem receives data from the measurement hardware and performs mathematical operations such as averaging, limit testing, and scaling before the data is stored in reading memory or displayed.

7.1 CALCulate:AVERage:ALL?

Description: Queries the statistics of all channels in the scan list. Returns the Minimum, Maximum, Average, and Count for each channel.

Syntax:
Query `CALCulate:AVERage:ALL?`
Parameters: None
Example:
Query `CALC:AVER:ALL?`

NOTICE **The returned format is: <Min_Ch1>,<Max_Ch1>,<Avg_Ch1>,<Cnt_Ch1>, <Min_Ch2>...**

7.2 CALCulate:AVERage:AVERage?

Description: Queries the calculated average value of the measurements for the specified channel(s).

Syntax:
Query `CALCulate:AVERage:AVERage? (@<ch_list>)`
Parameters: `<@ch_list>` : (@101:105)
Example:
Query `CALC:AVER:AVER? (@101)`

7.3 CALCulate:AVERage:CLEar

Description: Clears the statistical data (Min, Max, Average, Count) for the specified channel(s). If no channel is specified, it clears statistics for all channels.

Syntax:
Command `CALCulate:AVERage:CLEar [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@101)
Example:
Command `CALC:AVER:CLE (@101)`

7.4 CALCulate:AVERage:COUNt?

Description: Queries the number of measurements taken for the specified channel(s) since the last clear.

Syntax:
Query `CALCulate:AVERage:COUNt? (@<ch_list>)`
Parameters: `<@ch_list>` : (@101)
Example:
Query `CALC:AVER:COUN? (@101)`

7.5 CALCulate:AVERage:MAXimum?

Description: Queries the maximum value measured for the specified channel(s).

Syntax:
Query `CALCulate:AVERage:MAXimum? (@<ch_list>)`
Parameters: `<@ch_list>` : (@101)
Example:
Query `CALC:AVER:MAX? (@101)`

7.6 CALCulate:AVERage:MINimum?

Description: Queries the minimum value measured for the specified channel(s).

Syntax:
Query `CALCulate:AVERage:MINimum? (@<ch_list>)`
Parameters: `<@ch_list>` : (@101)
Example:
Query `CALC:AVER:MIN? (@101)`

7.7 CALCulate:LIMit:LOWer

Description: Sets or queries the lower limit for limit testing. When a measurement falls below this value, a limit violation occurs.

Syntax:
Command `CALCulate:LIMit:LOWer <value> [,@<ch_list>]`
Query `CALCulate:LIMit:LOWer? [(@<ch_list>)]`
Parameters: `<value>` : -1.0E+9 to 1.0E+9 `<@ch_list>` : (@101)
Example:
Command `CALC:LIM:LOW 4.5,(@101)`
Query `CALC:LIM:LOW? (@101)`

7.8 CALCulate:LIMit:UPPer

Description: Sets or queries the upper limit for limit testing. When a measurement exceeds this value, a limit violation occurs.

Syntax:
Command `CALCulate:LIMit:UPPer <value> [,@<ch_list>]`
Query `CALCulate:LIMit:UPPer? [(@<ch_list>)]`
Parameters: `<value>` : -1.0E+9 to 1.0E+9 `<@ch_list>` : (@101)
Example:
Command `CALC:LIM:UPP 5.5,(@101)`
Query `CALC:LIM:UPP? (@101)`

7.9 CALCulate:LIMit:STATe

Description: Enables or disables limit testing for the specified channel(s). When enabled, the instrument checks readings against the set Lower and Upper limits.

Syntax:
Command `CALCulate:LIMit:STATe <Boolean> [,@<ch_list>]`
Query `CALCulate:LIMit:STATe? [(@<ch_list>)]`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `CALC:LIM:STAT ON,(@101)`
Query `CALC:LIM:STAT? (@101)`

7.10 CALCulate:SCALe:GAIN

Description: Sets or queries the gain (slope) for the Mx+B scaling function. The measured value is multiplied by this gain before the offset is added.

Syntax:

Command `CALCulate:SCALe:GAIN <value> [,@<ch_list>]`
Query `CALCulate:SCALe:GAIN? [(@<ch_list>)]`

Parameters: `<value>` : -1.0E+9 to 1.0E+9 `<@ch_list>` : (@101)

Example:

Command `CALC:SCAL:GAIN 1.2,(@101)`
Query `CALC:SCAL:GAIN? (@101)`

7.11 CALCulate:SCALe:OFFSet

Description: Sets or queries the offset (B) for the Mx+B scaling function. This value is added to the result after the gain multiplication.

Syntax:

Command `CALCulate:SCALe:OFFSet <value> [,@<ch_list>]`
Query `CALCulate:SCALe:OFFSet? [(@<ch_list>)]`

Parameters: `<value>` : -1.0E+9 to 1.0E+9 `<@ch_list>` : (@101)

Example:

Command `CALC:SCAL:OFFS 0.5,(@101)`
Query `CALC:SCAL:OFFS? (@101)`

7.12 CALCulate:SCALe:STATe

Description: Enables or disables the Mx+B scaling function for the specified channel(s). When enabled, the displayed reading is (Measurement * Gain) + Offset.

Syntax:

Command `CALCulate:SCALe:STATe <Boolean> [,@<ch_list>]`
Query `CALCulate:SCALe:STATe? [(@<ch_list>)]`

Parameters: `<Boolean>` : 0 | 1 | OFF | ON

Example:

Command `CALC:SCAL:STAT ON,(@101)`
Query `CALC:SCAL:STAT? (@101)`

7.13 CALCulate:SMOothing:RESPonse

Description: Sets or queries the response speed of the smoothing filter (Fast, Medium, Slow). This affects how quickly the filter adapts to changes in the input signal.

Syntax:
Command `CALCulate:SMOothing:RESPonse <mode>`
Query `CALCulate:SMOothing:RESPonse?`
Parameters: `<mode>` : FAST | MEDium | SLOW
Example:
Command `CALC:SMO:RESP SLOW`
Query `CALC:SMO:RESP?`

7.14 CALCulate:SMOothing:STATe

Description: Enables or disables the smoothing digital filter. Smoothing helps reduce noise in measurements by averaging multiple samples.

Syntax:
Command `CALCulate:SMOothing:STATe <Boolean>`
Query `CALCulate:SMOothing:STATe?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `CALC:SMO:STAT ON`
Query `CALC:SMO:STAT?`

Configure Subsystem

The Configure subsystem provides a simplified way to program the instrument for measurements. Executing a `CONFigure` command sets up the specified function with default parameters (like range and resolution) for the given channels, but it does not initiate the measurement.

8.1 CONFigure?

Description: Queries the present configuration of the specified channel. Returns a quoted string indicating the function, range, and resolution.

Syntax:
Query `CONFigure? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@101)
Example:
Query `CONF? (@101)`

NOTICE **Example Return: "VOLT +1.000000E+01,+3.000000E-06"**

8.2 CONFigure:CAPacitance

Description: Configures the specified channels for capacitance measurements.

Syntax:
Command `CONFigure:CAPacitance [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (1nF to 100uF)
`<resolution>` : <value> | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Command `CONF:CAP AUTO,(@101)`

8.3 CONFigure:CURRent:AC|DC

Description: Configures the specified channels for AC or DC current measurements. Sets the range and resolution to defaults or user-specified values.

Syntax:
Command `CONFigure:CURRent:AC|DC [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (10mA to 1A)
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@121)
Example:
Command `CONF:CURR:DC 1A,(@121)`

8.4 CONFigure:DAC:OUTPut

Description: Configures the specified channels on a multifunction module as analog voltage outputs.

Syntax:

Command `CONFigure:DAC:OUTPut [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@201)

Example:

Command `CONF:DAC:OUTP (@201)`

8.5 CONFigure:DAC:SENSe

Description: Configures the specified DAC channels to verify the output level (sense mode).

Syntax:

Command `CONFigure:DAC:SENSe [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@201)

Example:

Command `CONF:DAC:SENS (@201)`

8.6 CONFigure:DIGital[:BYTE]

Description: Configures the digital I/O channels as a byte-wide input or output. This sets the width of the digital operation.

Syntax:

Command `CONFigure:DIGital[:BYTE] [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@201)

Example:

Command `CONF:DIG (@201)`

8.7 CONFigure:DIODe

Description: Configures the specified channels for diode testing. The instrument forces a current through the diode and measures the voltage drop.

Syntax:
Command `CONFigure:DIODe [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@101)
Example:
Command `CONF:DIOD (@101)`

8.8 CONFigure:FREQuency|PERiod

Description: Configures the specified channels for frequency or period measurements. The input signal must be AC voltage.

Syntax:
Command `CONFigure:FREQuency|PERiod [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | MIN | MAX | DEF (Auto range only, value ignored)
`<resolution>` : <value> | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Command `CONF:FREQ (@101)`

8.9 CONFigure:RESistance|FRESistance

Description: Configures the specified channels for 2-wire (RESistance) or 4-wire (FRESistance) resistance measurements.

Syntax:
Command `CONFigure:RESistance|FRESistance [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (100Ω to 100MΩ)
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Command `CONF:RES 10k,(@101)`

8.10 CONFigure:STRain:DIRect|FDIRect

Description: Configures the specified channels for Direct (2-wire) or Full-Direct (4-wire) strain measurements, assuming an external bridge.

Syntax:
Command `CONFigure:STRain:DIRect|FDIRect [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:
Command `CONF:STR:DIR (@101)`

8.11 CONFigure:STRain:FULL|HALF:BENDing

Description: Configures the specified channels for Full or Half bridge strain measurements specifically for Bending Strain.

Syntax:
Command `CONFigure:STRain:FULL|HALF:BENDing [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:
Command `CONF:STR:FULL:BEND (@101)`

8.12 CONFigure:STRain:FULL|HALF:POISson

Description: Configures the specified channels for Full or Half bridge strain measurements using Poisson ratio calculations.

Syntax:
Command `CONFigure:STRain:FULL|HALF:POISson [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:
Command `CONF:STR:HALF:POIS (@101)`

8.13 CONFigure:STRain:FULL:BENDing:POISson

Description: Configures the specified channels for Full bridge bending strain measurements with Poisson ratio compensation.

Syntax:

Command `CONFigure:STRain:FULL:BENDing:POISson [<range> [,<resolution>]] [,@<ch_lis`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:

Command `CONF:STR:FULL:BEND:POIS (@101)`

8.14 CONFigure:STRain:QUARter

Description: Configures the specified channels for Quarter bridge strain measurements.

Syntax:

Command `CONFigure:STRain:QUARter [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:

Command `CONF:STR:QUAR (@101)`

8.15 CONFigure:TEMPerature

Description: Configures the specified channels for temperature measurements using Thermocouples, RTDs, or Thermistors.

Syntax:

Command `CONFigure:TEMPerature [<probe_type> [,<type> [,1 [,<resolution>]]]] [,@<ch_list>]`

Parameters: `<probe_type>` : FRTD | RTD | FTH | THER | TC
`<type>` : J, K, T (for TC) | PT100 (for RTD), etc.
`<resolution>` : <value> | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:

Command `CONF:TEMP TC,K,(@101)`

NOTICE **For TCs, the type parameter specifies the thermocouple type (B, E, J, K, N, R, S, T).**

8.16 CONFigure:TOTalize

Description: Configures the specified channels as a digital totalizer. Counts the number of events (edges) on the input.

Syntax:

Command `CONFigure:TOTalize <mode> [,@<ch_list>]`

Parameters: `<mode>` : READ (Read only) | RRESet (Read and Reset)
`<@ch_list>` : (@201)

Example:

Command `CONF:TOT READ,(@201)`

8.17 CONFigure[:VOLTage]:AC|DC

Description: Configures the specified channels for AC or DC voltage measurements.

Syntax:

Command `CONFigure[:VOLTage]:AC|DC [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (100mV to 300V)
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)

Example:

Command `CONF:VOLT:DC 10,(@101)`

Data Subsystem

The Data subsystem commands allow you to manage the reading memory. You can query the number of readings stored, read the most recent measurement, or remove specific readings from the memory buffer.

9.1 DATA:LAST?

Description: Queries the last measurement taken. This is useful for monitoring the progress of a scan without interrupting it. This command does not remove data from memory.

Syntax:
Query `DATA:LAST? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@101)
Example:
Query `DATA:LAST? (@101)`

NOTICE **If specific channels are not provided, it returns the last reading taken from the scan list.**

9.2 DATA:POINts?

Description: Queries the total number of readings currently stored in the instrument's reading memory.

Syntax:
Query `DATA:POINts?`
Parameters: None
Example:
Query `DATA:POIN?`

NOTICE **The reading memory can hold up to 100,000 readings. Use this command to check for memory overflow conditions.**

9.3 DATA:POINts:EVENt:THReshold

Description: Sets or queries the number of readings required to generate a "Memory Threshold" event in the Standard Operation Status Group.

Syntax:
Command `DATA:POINts:EVENt:THReshold <count>`
Query `DATA:POINts:EVENt:THReshold?`
Parameters: `<count>` : 1 to 100,000
Example:

Command `DATA:POIN:EVEN:THR 500`
Query `DATA:POIN:EVEN:THR?`

9.4 DATA:REMove?

Description: Reads and erases the specified number of readings from the reading memory, starting with the oldest data. This allows for continuous data retrieval during long scans.

Syntax:
Query `DATA:REMove? <count>`
Parameters: `<count>` : 1 to 100,000
Example:
Query `DATA:REM? 10`

NOTICE **This command is similar to the `R?` command but adheres to standard SCPI syntax.**

Digital Interface Subsystem

The Digital Interface Subsystem commands configure the rear-panel Digital I/O port. This port can be used for general-purpose I/O, controlling external relays, or interfacing with handlers using specific protocols.

10.1 DIGital:INTerface:MODE

Description: Sets or queries the application mode of the rear-panel digital I/O port.

- **IO:** General purpose Input/Output mode.
- **4094:** Serial-to-Parallel mode (e.g., for driving strobed shift registers).
- **COPM:** Component Handler interface mode.

Syntax:
Command `DIGital:INTerface:MODE <type>`
Query `DIGital:INTerface:MODE?`
Parameters: `<type>` : IO | 4094 | COPM
Example:
Command `DIG:INT:MODE IO`
Query `DIG:INT:MODE?`

10.2 DIGital:INTerface:DATA:OUTPut

Description: Sets the output data and strobe pulse when the interface is in **4094** mode. This sends serial data to external shift registers.

Syntax:
Command `DIGital:INTerface:DATA:OUTPut <data>,<strobe>`

Parameters: `<data>` : 0 to 255 (8-bit value) `<strobe>` : 0 | 1
Example:
Command `DIG:INT:DATA:OUTP 10,1`

10.3 DIGital:INTerface:DATA:SETup

Description: Sets the logic state of the 4 digital output lines when the interface is in **IO** mode.

Syntax:
Command `DIGital:INTerface:DATA:SETup <bit1>,<bit2>,<bit3>,<bit4>`

Parameters: `<bit>` : 0 | 1 (Low | High)
Example:
Command `DIG:INT:DATA:SET 0,1,0,1`

NOTICE **This sets Line 1 Low, Line 2 High, Line 3 Low, and Line 4 High.**

Display Subsystem

The Display subsystem controls the instrument's front-panel display. You can turn the display on or off, write text to the screen, or clear the displayed text.

11.1 DISPlay

Description: Turns the front-panel display on or off. Turning the display off speeds up command processing by ignoring the display overhead. The display is automatically enabled when power is cycled or the instrument is reset (*RST).

Syntax:

Command `DISPlay <Boolean>`

Query `DISPlay?`

Parameters: `<Boolean>` : 0 | 1 | OFF | ON

Example:

Command `DISP OFF`

Query `DISP?`

11.2 DISPlay:TEXT

Description: Displays a text string on the front panel. This can be used to show messages or instructions to the operator. The message overrides the standard display until cleared.

Syntax:

Command `DISPlay:TEXT <string>`

Query `DISPlay:TEXT?`

Parameters: `<string>` : Quoted string (e.g., "TESTING")

Example:

Command `DISP:TEXT "STANDBY"`

Query `DISP:TEXT?`

NOTICE **The display can show up to 12 characters. Longer strings may be truncated.**

11.3 DISPlay:TEXT:CLEar

Description: Clears the text message displayed by the `DISPlay:TEXT` command and returns the display to its normal measurement mode.

Syntax:

Command `DISPlay:TEXT:CLEar`

Parameters: None

Example:

Command `DISP:TEXT:CLE`

Format Subsystem

The Format subsystem controls the data format of the response returned by the instrument. These commands determine whether additional information, such as channel numbers, time stamps, and engineering units, is included with the measurement data string.

12.1 FORMat:READing:ALARm

Description: Enables or disables the inclusion of alarm status information in the data string. When enabled, a value indicating the alarm state (0=No Alarm, 1=Low, 2=High, etc.) is returned with each reading.

Syntax:
Command `FORMat:READing:ALARm <Boolean>`
Query `FORMat:READing:ALARm?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `FORM:READ:ALAR ON`
Query `FORM:READ:ALAR?`

12.2 FORMat:READing:CHANnel

Description: Enables or disables the inclusion of the channel number in the data string. When enabled, the channel number (e.g., 101) is returned preceding the measurement value.

Syntax:
Command `FORMat:READing:CHANnel <Boolean>`
Query `FORMat:READing:CHANnel?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `FORM:READ:CHAN ON`
Query `FORM:READ:CHAN?`

12.3 FORMat:READing:DATE

Description: Enables or disables the inclusion of the date timestamp in the data string. When enabled, the date (YYYY,MM,DD) is returned with each reading.

Syntax:
Command `FORMat:READing:DATE <Boolean>`
Query `FORMat:READing:DATE?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `FORM:READ:DATE ON`
Query `FORM:READ:DATE?`

12.4 FORMat:READing:TIME

Description: Enables or disables the inclusion of the time timestamp in the data string. When enabled, the time (HH,MM,SS.sss) is returned with each reading.

Syntax:
Command `FORMat:READing:TIME <Boolean>`
Query `FORMat:READing:TIME?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `FORM:READ:TIME ON`
Query `FORM:READ:TIME?`

12.5 FORMat:READing:TIME:TYPE

Description: Sets or queries the format of the timestamp.

- **ABS:** Absolute time (Calendar date and time).
- **REL:** Relative time (Time elapsed since the start of the scan).

Syntax:
Command `FORMat:READing:TIME:TYPE <type>`
Query `FORMat:READing:TIME:TYPE?`
Parameters: `<type>` : ABS | REL
Example:
Command `FORM:READ:TIME:TYPE REL`
Query `FORM:READ:TIME:TYPE?`

12.6 FORMat:READing:UNIT

Description: Enables or disables the inclusion of the measurement unit (e.g., VDC, OHM, C) in the data string.

Syntax:
Command `FORMat:READing:UNIT <Boolean>`
Query `FORMat:READing:UNIT?`

Parameters: <Boolean> : 0 | 1 | OFF | ON

Example:

Command `FORM:READ:UNIT OFF`

Query `FORM:READ:UNIT?`

HCOPy Subsystem

The HCOPy (Hard Copy) subsystem commands allow you to capture the current image of the front panel display. This is useful for documentation, remote monitoring, and troubleshooting.

13.1 HCOPy:SDUMp:DATA?

Description: Captures the front panel screen and returns the data as a definite-length binary block. The format of the image is determined by the `HCOPy:SDUMp:FORMat` command.

Syntax:
Query `HCOPy:SDUMp:DATA?`
Parameters: None
Example:
Query `HCOP:SDUM:DATA?`

NOTICE **The response is a binary block (e.g., #524036...). You must read the block header to determine the size of the image data.**

13.2 HCOPy:SDUMp:FORMat

Description: Sets or queries the file format used for the screen dump data.

Syntax:
Command `HCOPy:SDUMp:FORMat BMP|PNG`
Query `HCOPy:SDUMp:FORMat?`
Parameters: `<format>` : BMP (Bitmap) | PNG (Portable Network Graphics)
Example:
Command `HCOP:SDUM:FORM PNG`
Query `HCOP:SDUM:FORM?`

Measure Subsystem

The Measure subsystem provides the easiest way to program the instrument for measurements. Executing a `MEASure?` command configures the specified function with default or user-specified parameters, initiates the scan immediately, and returns the measurement result.
It is essentially a macro that performs: `ABORt` + `CONFigure` + `READ?`.

14.1 MEASure?

Description: Configures and measures the specified channels using their present function settings. If no channel list is provided, it measures the currently selected channel.

Syntax:
Query `MEASure? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@101)
Example:
Query `MEAS? (@101)`

14.2 MEASure:CAPacitance?

Description: Configures the specified channels for capacitance measurements and immediately triggers a measurement.

Syntax:
Query `MEASure:CAPacitance? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (1nF to 100uF)
`<resolution>` : <value> | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:CAP? (@101)`

14.3 MEASure:CURRent:AC|DC?

Description: Configures the specified channels for AC or DC current measurements and immediately triggers a measurement.

Syntax:
Query `MEASure:CURRent:AC|DC? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (10mA to 1A)
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@121)
Example:
Query `MEAS:CURR:DC? 100mA,(@121)`

14.4 MEASure:DAC:OUTPut?

Description: Configures and measures the output voltage level of the specified DAC channels on a multifunction module.

Syntax:
Query `MEASure:DAC:OUTPut? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@201)
Example:
Query `MEAS:DAC:OUTP? (@201)`

14.5 MEASure:DAC:SENSe?

Description: Configures and measures the voltage sensed at the load for the specified DAC channels (using Sense lines).

Syntax:
Query `MEASure:DAC:SENSe? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@201)
Example:
Query `MEAS:DAC:SENS? (@201)`

14.6 MEASure:DIGital[:BYTE]?

Description: Configures and reads the digital control port as a byte-wide input. Returns the integer value of the 8-bit digital input.

Syntax:
Query `MEASure:DIGital[:BYTE]? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@201)
Example:
Query `MEAS:DIG? (@201)`

14.7 MEASure:DIODe?

Description: Configures the specified channels for diode testing and immediately returns the measured voltage drop.

Syntax:
Query `MEASure:DIODe? [(@<ch_list>)]`
Parameters: `<@ch_list>` : (@101)
Example:
Query `MEAS:DIOD? (@101)`

14.8 MEASure:FREQuency|PERiod?

Description: Configures the specified channels for frequency or period measurements and immediately triggers a measurement.

Syntax:
Query `MEASure:FREQuency|PERiod? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : ‹value› | MIN | MAX | DEF (Auto range only, value ignored)
`<resolution>` : ‹value› | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:FREQ? (@101)`

14.9 MEASure:RESistance|FRESistance?

Description: Configures the specified channels for 2-wire (RESistance) or 4-wire (FRESistance) resistance measurements and immediately returns the value.

Syntax:
Query `MEASure:RESistance|FRESistance? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF (100Ω to 100MΩ)
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:RES? 10k,(@101)`

14.10 MEASure:STRain:DIRect|FDIRect?

Description: Configures and measures Direct (2-wire) or Full-Direct (4-wire) strain from the specified channels.

Syntax:
Query `MEASure:STRain:DIRect|FDIRect? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:STR:DIR? (@101)`

14.11 MEASure:STRain:FULL|HALF:BENDing?

Description: Configures and measures Bending Strain using a Full or Half bridge configuration on the specified channels.

Syntax:
Query `MEASure:STRain:FULL|HALF:BENDing? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:STR:FULL:BEND? (@101)`

14.12 MEASure:STRain:FULL|HALF:POISson?

Description: Configures and measures strain using Poisson ratio calculations with a Full or Half bridge configuration.

Syntax:
Query `MEASure:STRain:FULL|HALF:POISson? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : ‹value› | AUTO | MIN | MAX | DEF
`<resolution>` : ‹value› | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:STR:HALF:POIS? (@101)`

14.13 MEASure:STRain:FULL:BENDing:POISson?

Description: Configures and measures Full bridge bending strain with Poisson ratio compensation.

Syntax:
Query `MEASure:STRain:FULL:BENDing:POISson? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:STR:FULL:BEND:POIS? (@101)`

14.14 MEASure:STRain:QUARter?

Description: Configures and measures strain using a Quarter bridge configuration on the specified channels.

Syntax:
Query `MEASure:STRain:QUARter? [<range> [,<resolution>]] [,@<ch_list>]`
Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF
`<resolution>` : <value> | AUTO | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:STR:QUAR? (@101)`

14.15 MEASure:TEMPerature?

Description: Configures the specified channels for temperature measurements and immediately returns the result.

Syntax:
Query `MEASure:TEMPerature? [<probe_type> [,<type> [,1 [,<resolution>]]]] [,@<ch_list`
Parameters: `<probe_type>` : FRTD | RTD | FTH | THER | TC
`<type>` : J, K, T (for TC) | PT100 (for RTD), etc.
`<resolution>` : <value> | MIN | MAX | DEF
`<@ch_list>` : (@101)
Example:
Query `MEAS:TEMP? TC,K,(@101)`

14.16 MEASure:TOTalize?

Description: Configures and reads the count from the totalizer channels.

Syntax:

Query `MEASure:TOTalize? <mode> [,@<ch_list>]`

Parameters: `<mode>` : READ | RRESet

`<@ch_list>` : (@201)

Example:

Query `MEAS:TOT? READ,(@201)`

14.17 MEASure[:VOLTage]:AC|DC?

Description: Configures the specified channels for AC or DC voltage measurements and immediately triggers a measurement.

Syntax:

Query `MEASure[:VOLTage]:AC|DC? [<range> [,<resolution>]] [,@<ch_list>]`

Parameters: `<range>` : <value> | AUTO | MIN | MAX | DEF (100mV to 300V)

`<resolution>` : <value> | AUTO | MIN | MAX | DEF

`<@ch_list>` : (@101)

Example:

Query `MEAS:VOLT:DC? 10,(@101)`

MMemory Subsystem

The MMemory (Mass Memory) subsystem commands configure the format of data stored in the instrument's reading memory and control data logging to the USB drive.

15.1 MMEMory:FORMat:READing:CHEAder

Description: Specifies whether the column header for saved data is the channel number or the user-defined channel label.

- **NUMber:** Uses the channel number (e.g., 101).
- **LABel:** Uses the user-defined label. If no label is set, the default channel label is used.

Syntax:
Command `MMEMory:FORMat:READing:CHEAder NUMber|LABel`
Query `MMEMory:FORMat:READing:CHEAder?`
Parameters: `<format>` : NUMber | LABel
Example:
Command `MMEM:FORM:READ:CHEA LAB`
Query `MMEM:FORM:READ:CHEA?`

15.2 MMEMory:FORMat:READing:CSEParator

Description: Specifies the character used to separate columns in the saved data file (CSV format).

Syntax:
Command `MMEMory:FORMat:READing:CSEParator COMMa|TAB|SEMicolon`
Query `MMEMory:FORMat:READing:CSEParator?`
Parameters: `<separator>` : COMMa (,) | TAB (\t) | SEMicolon (;)
Example:
Command `MMEM:FORM:READ:CSEP COMM`
Query `MMEM:FORM:READ:CSEP?`

15.3 MMEMory:FORMat:READing:RLIMit

Description: Enables or disables the row limit for data logging. When enabled, the instrument will stop logging or start a new file after a specified number of rows.

Syntax:
Command `MMEMory:FORMat:READing:RLIMit <Boolean>`
Query `MMEMory:FORMat:READing:RLIMit?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:

Command `MMEM:FORM:READ:RLIM ON`
Query `MMEM:FORM:READ:RLIM?`

15.4 MMEMory:FORMat:READing:RLIMit:COUNt

Description: Sets or queries the maximum number of rows (readings) per file when row limiting is enabled.

Syntax:
Command `MMEMory:FORMat:READing:RLIMit:COUNt <count>`
Query `MMEMory:FORMat:READing:RLIMit:COUNt?`
Parameters: `<count>` : 1 to 1,000,000
Example:
Command `MMEM:FORM:READ:RLIM:COUN 1000`
Query `MMEM:FORM:READ:RLIM:COUN?`

15.5 MMEMory:LOG[:ENABle]

Description: Enables or disables data logging to the connected USB drive. When enabled, scan data is automatically written to the USB drive.

Syntax:
Command `MMEMory:LOG[:ENABle] <Boolean>`
Query `MMEMory:LOG[:ENABle]?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `MMEM:LOG ON`
Query `MMEM:LOG?`

Output Subsystem

The Output subsystem controls the four rear-panel alarm output lines. These lines can be used to trigger external devices (like lights, sirens, or shut-off switches) when a limit violation occurs on a channel.

16.1 OUTPut:ALARm:CLEar

Description: Clears the state of the four alarm output lines. This turns off the alarm output even if the alarm condition still exists (if the mode is set to Latch).

Syntax:
Command `OUTPut:ALARm:CLEar`
Query `OUTPut:ALARm:CLEar?`
Parameters: None
Example:
Command `OUTP:ALAR:CLE`

NOTICE **This command allows the operator to acknowledge an alarm condition without stopping the scan.**

16.2 OUTPut:ALARm:MODE

Description: Sets or queries the operating mode of the four alarm output lines.

- **LATCh:** The alarm output remains asserted until explicitly cleared by `OUTPut:ALARm:CLEar` or a reset.
- **TRACk:** The alarm output follows the state of the alarm condition. It clears automatically when the reading returns to within limits.

Syntax:
Command `OUTPut:ALARm:MODE LATCh|TRACk`
Query `OUTPut:ALARm:MODE?`
Parameters: `<mode>` : LATCh | TRACk
Example:
Command `OUTP:ALAR:MODE LATC`
Query `OUTP:ALAR:MODE?`

16.3 OUTPut:ALARm:SLOPe

Description: Sets or queries the logic polarity of the four alarm output lines.

- **NEGative:** The line is asserted Low (TTL Logic 0) when an alarm occurs.

- **POSitive:** The line is asserted High (TTL Logic 1) when an alarm occurs.

Syntax:

Command `OUTPut:ALARm:SLOPe NEGative|POSitive`

Query `OUTPut:ALARm:SLOPe?`

Parameters: `<slope>` : NEGative | POSitive

Example:

Command `OUTP:ALAR:SLOP NEG`

Query `OUTP:ALAR:SLOP?`

Route Subsystem

The Route subsystem controls the signal routing of the plug-in modules. These commands allow you to open or close individual relays, define a list of channels to be scanned, and configure the monitor function.

17.1 ROUTe:CLOSe

Description: Closes the specified channels on a multiplexer or switch module. This command allows for multiple channels to be closed simultaneously (e.g., for matrix operations).

Syntax:
Command `ROUTe:CLOSe (@<ch_list>)`
Query `ROUTe:CLOSe? (@<ch_list>)`
Parameters: `<@ch_list>` : (@101, 102)
Example:
Command `ROUT:CLOS (@101,102)`
Query `ROUT:CLOS? (@101)`

NOTICE **To close a channel while ensuring all others on the bank are open (typical multiplexer behavior), use `ROUTe:CLOSe:EXCLusive`.**

17.2 ROUTe:CLOSe:EXCLusive

Description: Closes the specified channel and opens all other channels on the same bank (Break-Before-Make). This ensures that only one channel in the bank is connected to the common bus at a time.

Syntax:
Command `ROUTe:CLOSe:EXCLusive (@<ch_list>)`
Parameters: `<@ch_list>` : (@101)
Example:
Command `ROUT:CLOS:EXCL (@101)`

17.3 ROUTe:DONE?

Description: Queries the completion status of a routing operation. Returns 1 if all relay operations are finished, allowing for synchronization.

Syntax:
Query `ROUTe:DONE?`
Parameters: None
Example:
Query `ROUT:DONE?`

17.4 ROUTe:MONitor

Description: Specifies the channel to be monitored. The instrument measures this channel continuously when the Monitor function is enabled.

Syntax:
Command `ROUTe:MONitor (@<ch_list>)`
Query `ROUTe:MONitor?`
Parameters: `<@ch_list>` : (@101)
Example:
Command `ROUT:MON (@101)`
Query `ROUT:MON?`

NOTICE **Only one channel can be monitored at a time.**

17.5 ROUTe:MONitor:STATe

Description: Enables or disables the Monitor mode. When enabled, the instrument continuously measures the channel specified by `ROUTe:MONitor`.

Syntax:
Command `ROUTe:MONitor:STATe <Boolean>`
Query `ROUTe:MONitor:STATe?`
Parameters: `<Boolean>` : 0 | 1 | OFF | ON
Example:
Command `ROUT:MON:STAT ON`
Query `ROUT:MON:STAT?`

17.6 ROUTe:OPEN

Description: Opens the specified channels on a switch module.

Syntax:

Command `ROUTe:OPEN (@<ch_list>)`
Query `ROUTe:OPEN? (@<ch_list>)`

Parameters: `<@ch_list>` : (@101)

Example:

Command `ROUT:OPEN (@101)`
Query `ROUT:OPEN? (@101)`

17.7 ROUTe:SCAN

Description: Defines the list of channels to be included in the scan. When the scan is initiated (e.g., by `INITiate` or `READ?`), the instrument cycles through these channels.

Syntax:

Command `ROUTe:SCAN (@<ch_list>)`
Query `ROUTe:SCAN?`

Parameters: `<@ch_list>` : (@101:105)

Example:

Command `ROUT:SCAN (@101:110)`
Query `ROUT:SCAN?`

NOTICE **To clear the scan list, send `ROUTe:SCAN (@)`.**

17.8 ROUTe:SCAN:SIZE?

Description: Queries the number of channels currently in the scan list.

Syntax:

Query `ROUTe:SCAN:SIZE?`

Parameters: None

Example:

Query `ROUT:SCAN:SIZE?`

Sense Subsystem

The Sense subsystem commands are used to configure the specific measurement parameters for each function (such as range, resolution, integration time, and transducer type) without immediately initiating a scan. These commands allow for fine-grained control over the measurement process compared to the higher-level `CONFigure` commands.

18.1 SENSe Related Commands

18.1.1 [SENSe:]FUNCtion[:ON]

Description: Selects the measurement function for the specified channels. This command overrides any previous function configuration for the selected channels.

Syntax:

Command `[SENSe:]FUNCtion[:ON] "<function>" [,@<ch_list>]`

Query `[SENSe:]FUNCtion[:ON]? [(@<ch_list>)]`

Parameters: `<function>`: "VOLT:DC", "VOLT:AC", "CURR:DC", "CURR:AC", "RES", "FRES", "FREQ", "PER", "TEMP", "CAP", "STR", "DIOD", "TOT"

`<@ch_list>`: (@101)

Example:

Command `FUNC "VOLT:DC",(@101)`

Query `FUNC? (@101)`

NOTICE **The function string must be enclosed in double quotes.**

18.2 SENSe AVERage Commands

18.2.1 [SENSe:]AVERage:COUNt

Description: Sets or queries the number of readings to be averaged for the digital filter.

Syntax:

Command `[SENSe:]AVERage:COUNt <count> [,@<ch_list>]`

Query `[SENSe:]AVERage:COUNt? [(@<ch_list>)]`

Parameters: `<count>`: 1 to 100

`<@ch_list>`: (@101)

Example:

Command `AVER:COUN 10,(@101)`

Query `AVER:COUN? (@101)`

18.2.2 [SENSe:]AVERage:STATe

Description: Enables or disables the digital averaging filter.

Syntax:

Command `[SENSe:]AVERage:STATe <state> [,@<ch_list>]`
Query `[SENSe:]AVERage:STATe? [(@<ch_list>)]`

Parameters: `<state>` : 0 | 1 | OFF | ON
`<@ch_list>` : (@101)

Example:

Command `AVER:STAT ON,(@101)`
Query `AVER:STAT? (@101)`

18.2.3 [SENSe:]AVERage:WINDow

Description: Sets the window size for the digital filter, usually as a percentage of the range.

Syntax:

Command `[SENSe:]AVERage:WINDow <value> [,@<ch_list>]`
Query `[SENSe:]AVERage:WINDow? [(@<ch_list>)]`

Parameters: `<value>` : 0 to 100 (Percent)
`<@ch_list>` : (@101)

Example:

Command `AVER:WIND 5,(@101)`
Query `AVER:WIND? (@101)`

18.2.4 [SENSe:]AVERage:WINDow:METHod

Description: Sets the method used for the averaging window (e.g., Repeating or Moving average).

Syntax:

Command `[SENSe:]AVERage:WINDow:METHod <method> [,@<ch_list>]`
Query `[SENSe:]AVERage:WINDow:METHod? [(@<ch_list>)]`

Parameters: `<method>` : REPeat | MOVe
`<@ch_list>` : (@101)

Example:

Command `AVER:WIND:METH MOV,(@101)`

18.3 SENSe CAPacitance Commands

18.3.1 [SENSe:]CAPacitance:RANGe

Description: Sets or queries the fixed measurement range for capacitance measurements.

Syntax:

Command `[SENSe:]CAPacitance:RANGe <range> [,@<ch_list>]`
Query `[SENSe:]CAPacitance:RANGe? [(@<ch_list>)]`

Parameters: `<range>` : <value> | MIN | MAX | DEF (1nF to 100uF)
`<@ch_list>` : (@101)

Example:

Command `CAP:RANG 10uF,(@101)`
Query `CAP:RANG? (@101)`

18.3.2 [SENSe:]CAPacitance:RANGe:AUTO

Description: Enables or disables auto-ranging for capacitance measurements.

Syntax:

Command `[SENSe:]CAPacitance:RANGe:AUTO <state> [,@<ch_list>]`
Query `[SENSe:]CAPacitance:RANGe:AUTO? [(@<ch_list>)]`

Parameters: `<state>` : OFF | ON | ONCE
`<@ch_list>` : (@101)

Example:

Command `CAP:RANG:AUTO ON,(@101)`

18.4 SENSe CURRent Commands

18.4.1 [SENSe:]CURRent:AC:BANDwidth

Description: Sets the bandwidth filter for AC current measurements.

Syntax:

Command `[SENSe:]CURRent:AC:BANDwidth <bandwidth> [,@<ch_list>]`
Query `[SENSe:]CURRent:AC:BANDwidth? [(@<ch_list>)]`

Parameters: `<bandwidth>` : 3 | 20 | 200 | MIN | MAX | DEF (Hz)

<ch_list> : (@121)

Example:

Command `CURR:AC:BAND 20,(@121)`

18.4.2 [SENSe:]CURRent:AC|DC:RANGe

Description: Sets the fixed measurement range for AC or DC current.

Syntax:

Command `[SENSe:]CURRent:AC:RANGe <range> [,@<ch_list>]`
`[SENSe:]CURRent:DC:RANGe <range> [,@<ch_list>]`
Query `[SENSe:]CURRent:AC:RANGe? [(@<ch_list>)]`
`[SENSe:]CURRent:DC:RANGe? [(@<ch_list>)]`

Parameters: <range> : <value> | MIN | MAX | DEF (10mA to 1A)

<ch_list> : (@121)

Example:

Command `CURR:DC:RANG 100mA,(@121)`

18.4.3 [SENSe:]CURRent:AC|DC:RANGe:AUTO

Description: Enables or disables auto-ranging for current measurements.

Syntax:

Command `[SENSe:]CURRent:AC:RANGe:AUTO <state> [,@<ch_list>]`
`[SENSe:]CURRent:DC:RANGe:AUTO <state> [,@<ch_list>]`
Query `[SENSe:]CURRent:AC:RANGe:AUTO? [(@<ch_list>)]`
`[SENSe:]CURRent:DC:RANGe:AUTO? [(@<ch_list>)]`

Parameters: <state> : OFF | ON | ONCE

<ch_list> : (@121)

Example:

Command `CURR:DC:RANG:AUTO ON,(@121)`

18.4.4 [SENSe:]CURRent:AC|DC:RANGe:LOW

Description: Sets the lowest range used during auto-ranging to optimize speed.

Syntax:

Command [SENSe:]CURRent:AC:RANGe:LOW <range> [,@<ch_list>]
[SENSe:]CURRent:DC:RANGe:LOW <range> [,@<ch_list>]
Query [SENSe:]CURRent:AC:RANGe:LOW? [(@<ch_list>)]
[SENSe:]CURRent:DC:RANGe:LOW? [(@<ch_list>)]

Parameters: <range> : <value> (Low limit)
<@ch_list> : (@121)

Example:
Command CURR:DC:RANG:LOW 10mA,(@121)

18.4.5 [SENSe:]CURRent[:DC]:APERture

Description: Sets the integration time in seconds.

Syntax:
Command [SENSe:]CURRent[:DC]:APERture <seconds> [,@<ch_list>]
Query [SENSe:]CURRent[:DC]:APERture? [(@<ch_list>)]

Parameters: <seconds> : <value> | MIN | MAX | DEF
<@ch_list> : (@121)

Example:
Command CURR:APER 0.1,(@121)

18.4.6 [SENSe:]CURRent[:DC]:APERture:ENABle

Description: Enables the definition of integration time by aperture (seconds) instead of NPLC.

Syntax:
Command [SENSe:]CURRent[:DC]:APERture:ENABle <state> [,@<ch_list>]
Query [SENSe:]CURRent[:DC]:APERture:ENABle? [(@<ch_list>)]

Parameters: <state> : 0 | 1 | OFF | ON
<@ch_list> : (@121)

Example:
Command CURR:APER:ENAB ON,(@121)

18.4.7 [SENSe:]CURRent[:DC]:NPLCycles

Description: Sets the integration time in Number of Power Line Cycles (NPLC).

Syntax:

Command [SENSe:]CURRent[:DC]:NPLCycles <PLCs> [,@<ch_list>]
Query [SENSe:]CURRent[:DC]:NPLCycles? [(@<ch_list>)]
Parameters: <PLCs> : <value> | MIN | MAX | DEF (0.02, 0.2, 1, 10, 20, 100, 200)
<@ch_list> : (@121)
Example:
Command CURR:NPLC 10,(@121)

18.4.8 [SENSe:]CURRent[:DC]:ZERO:AUTO

Description: Controls the auto-zero mode for DC current measurements.

Syntax:
Command [SENSe:]CURRent[:DC]:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]CURRent[:DC]:ZERO:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@121)
Example:
Command CURR:ZERO:AUTO ON,(@121)

18.5 SENSe DIODe Commands

18.5.1 [SENSe:]DIODe:ZERO:AUTO

Description: Controls the auto-zero mode for diode measurements.

Syntax:
Command [SENSe:]DIODe:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]DIODe:ZERO:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)
Example:
Command DIOD:ZERO:AUTO ON,(@101)

18.6 SENSe DIGital Commands

18.6.1 [SENSe:]DIGital:DATA[:BYTE]?

Description: Reads the digital I/O port as an 8-bit byte.

Syntax:

Query `[SENSe:]DIGital:DATA[:BYTE]? [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@201)

Example:

Query `DIG:DATA? (@201)`

18.6.2 [SENSe:]DIGital:DATA:WORD?

Description: Reads the digital I/O port as a 16-bit word (if supported by module).

Syntax:

Query `[SENSe:]DIGital:DATA:WORD? [(@<ch_list>)]`

Parameters: `<@ch_list>` : (@201)

Example:

Query `DIG:DATA:WORD? (@201)`

18.7 SENSe FREQuency Commands

18.7.1 [SENSe:]FREQuency|PERiod:APERture

Description: Sets the gate time (aperture) for frequency or period measurements.

Syntax:

Command `[SENSe:]FREQuency:APERture <seconds> [,@<ch_list>]`
`[SENSe:]PERiod:APERture <seconds> [,@<ch_list>]`

Query `[SENSe:]FREQuency:APERture? [(@<ch_list>)]`
`[SENSe:]PERiod:APERture? [(@<ch_list>)]`

Parameters: `<seconds>` : <value> | MIN | MAX | DEF (0.01s, 0.1s, 1s)
`<@ch_list>` : (@101)

Example:

Command `FREQ:APER 1,(@101)`

18.7.2 [SENSe:]FREQuency|PERiod:RANGe:LOWer

Description: Sets the lower bandwidth limit for the frequency signal filter.

Syntax:

Command `[SENSe:]FREQuency:RANGe:LOWer <filter> [,@<ch_list>]`
`[SENSe:]PERiod:RANGe:LOWer <filter> [,@<ch_list>]`
Query `[SENSe:]FREQuency:RANGe:LOWer? [(@<ch_list>)]`
`[SENSe:]PERiod:RANGe:LOWer? [(@<ch_list>)]`

Parameters: `<filter>` : <value> | MIN | MAX | DEF (3 Hz, 20 Hz, 200 Hz)
`<@ch_list>` : (@101)

Example:

Command `FREQ:RANG:LOW 20,(@101)`

18.7.3 [SENSe:]FREQuency|PERiod:TIMeout:AUTO

Description: Controls the timeout for frequency measurements. If the signal is too slow, the measurement will time out.

Syntax:

Command `[SENSe:]FREQuency:TIMeout:AUTO <state> [,@<ch_list>]`
`[SENSe:]PERiod:TIMeout:AUTO <state> [,@<ch_list>]`
Query `[SENSe:]FREQuency:TIMeout:AUTO? [(@<ch_list>)]`
`[SENSe:]PERiod:TIMeout:AUTO? [(@<ch_list>)]`

Parameters: `<state>` : OFF | ON
`<@ch_list>` : (@101)

Example:

Command `FREQ:TIM:AUTO ON,(@101)`

18.7.4 [SENSe:]FREQuency|PERiod:VOLTage:RANGe

Description: Sets the AC voltage range for the frequency input.

Syntax:

Command `[SENSe:]FREQuency:VOLTage:RANGe <range> [,@<ch_list>]`
`[SENSe:]PERiod:VOLTage:RANGe <range> [,@<ch_list>]`
Query `[SENSe:]FREQuency:VOLTage:RANGe? [(@<ch_list>)]`
`[SENSe:]PERiod:VOLTage:RANGe? [(@<ch_list>)]`

Parameters: `<range>` : <value> | MIN | MAX | DEF (100mV to 300V)
`<@ch_list>` : (@101)

Example:

Command `FREQ:VOLT:RANG 10,(@101)`

18.7.5 [SENSe:]FREQuency|PERiod:VOLTage:RANGe:AUTO

Description: Enables auto-ranging for the voltage input during frequency measurements.

Syntax:

Command `[SENSe:]FREQuency:VOLTage:RANGe:AUTO <state> [,@<ch_list>]`
`[SENSe:]PERiod:VOLTage:RANGe:AUTO <state> [,@<ch_list>]`
Query `[SENSe:]FREQuency:VOLTage:RANGe:AUTO? [(@<ch_list>)]`
`[SENSe:]PERiod:VOLTage:RANGe:AUTO? [(@<ch_list>)]`

Parameters: `<state>` : OFF | ON | ONCE
`<@ch_list>` : (@101)

Example:

Command `FREQ:VOLT:RANG:AUTO ON,(@101)`

18.8 SENSe RESistance Commands

18.8.1 [SENSe:]RESistance|FRESistance:APERture

Description: Sets the integration time in seconds for resistance measurements.

Syntax:

Command `[SENSe:]RESistance:APERture <seconds> [,@<ch_list>]`
`[SENSe:]FRESistance:APERture <seconds> [,@<ch_list>]`
Query `[SENSe:]RESistance:APERture? [(@<ch_list>)]`
`[SENSe:]FRESistance:APERture? [(@<ch_list>)]`

Parameters: `<seconds>` : <value> (Integration time)
`<@ch_list>` : (@101)

Example:

Command `RES:APER 0.1,(@101)`

18.8.2 [SENSe:]RESistance|FRESistance:APERture:ENABle

Description: Enables definition of integration time by aperture instead of NPLC.

Syntax:

Command `[SENSe:]RESistance:APERture:ENABle <state> [,@<ch_list>]`
`[SENSe:]FRESistance:APERture:ENABle <state> [,@<ch_list>]`
Query `[SENSe:]RESistance:APERture:ENABle? [(@<ch_list>)]`
`[SENSe:]FRESistance:APERture:ENABle? [(@<ch_list>)]`

Parameters: `<state>` : 0 | 1 | OFF | ON
`<@ch_list>` : (@101)

Example:

Command `RES:APER:ENAB ON,(@101)`

18.8.3 [SENSe:]RESistance|FRESistance:NPLCycles

Description: Sets the integration time in NPLC for resistance measurements.

Syntax:

Command `[SENSe:]RESistance:NPLCycles <PLCs> [,@<ch_list>]`
`[SENSe:]FRESistance:NPLCycles <PLCs> [,@<ch_list>]`
Query `[SENSe:]RESistance:NPLCycles? [(@<ch_list>)]`
`[SENSe:]FRESistance:NPLCycles? [(@<ch_list>)]`

Parameters: `<PLCs>` : ‹value› (0.02 to 200)
`<@ch_list>` : (@101)

Example:

Command `RES:NPLC 10,(@101)`

18.8.4 [SENSe:]RESistance|FRESistance:OCOMpensated

Description: Enables Offset Compensation for resistance measurements to cancel thermal EMF.

Syntax:

Command `[SENSe:]RESistance:OCOMpensated <state> [,@<ch_list>]`
`[SENSe:]FRESistance:OCOMpensated <state> [,@<ch_list>]`
Query `[SENSe:]RESistance:OCOMpensated? [(@<ch_list>)]`
`[SENSe:]FRESistance:OCOMpensated? [(@<ch_list>)]`

Parameters: <state> : OFF | ON
<@ch_list> : (@101)
Example:
Command RES:OCOM ON,(@101)

18.8.5 [SENSe:]RESistance|FRESistance:POWer:LIMit[:STATe]

Description: Enables low-power mode for resistance measurements to prevent self-heating.

Syntax:

Command [SENSe:]RESistance:POWer:LIMit[:STATe] <state> [,@<ch_list>]
[SENSe:]FRESistance:POWer:LIMit[:STATe] <state> [,@<ch_list>]
Query [SENSe:]RESistance:POWer:LIMit[:STATe]? [(@<ch_list>)]
[SENSe:]FRESistance:POWer:LIMit[:STATe]? [(@<ch_list>)]

Parameters: <state> : OFF | ON
<@ch_list> : (@101)
Example:
Command RES:POW:LIM ON,(@101)

18.8.6 [SENSe:]RESistance|FRESistance:RANGe

Description: Sets the fixed measurement range for resistance.

Syntax:

Command [SENSe:]RESistance:RANGe <range> [,@<ch_list>]
[SENSe:]FRESistance:RANGe <range> [,@<ch_list>]
Query [SENSe:]RESistance:RANGe? [(@<ch_list>)]
[SENSe:]FRESistance:RANGe? [(@<ch_list>)]

Parameters: <range> : <value> (100Ω to 100MΩ)
<@ch_list> : (@101)
Example:
Command RES:RANG 1000,(@101)

18.8.7 [SENSe:]RESistance|FRESistance:RANGe:AUTO

Description: Enables auto-ranging for resistance measurements.

Syntax:

Command [SENSe:]RESistance:RANGe:AUTO <state> [,@<ch_list>]
[SENSe:]FRESistance:RANGe:AUTO <state> [,@<ch_list>]
Query [SENSe:]RESistance:RANGe:AUTO? [(@<ch_list>)]
[SENSe:]FRESistance:RANGe:AUTO? [(@<ch_list>)]

Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)

Example:
Command RES:RANG:AUTO ON,(@101)

18.8.8 [SENSe:]RESistance|FRESistance:ZERO:AUTO

Description: Controls the auto-zero mode for resistance measurements.

Syntax:
Command [SENSe:]RESistance:ZERO:AUTO <state> [,@<ch_list>]
[SENSe:]FRESistance:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]RESistance:ZERO:AUTO? [(@<ch_list>)]
[SENSe:]FRESistance:ZERO:AUTO? [(@<ch_list>)]

Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)

Example:
Command RES:ZERO:AUTO ON,(@101)

18.9 SENSe STRain Commands

18.9.1 [SENSe:]STRain:APERture

Description: Sets the integration time in seconds for strain measurements.

Syntax:
Command [SENSe:]STRain:APERture <seconds> [,@<ch_list>]
Query [SENSe:]STRain:APERture? [(@<ch_list>)]

Parameters: <seconds> : <value> (Integration time)
<@ch_list> : (@101)

Example:
Command STR:APER 0.1,(@101)

18.9.2 [SENSe:]STRain:APERture:ENABle

Description: Enables definition of integration time by aperture for strain measurements.

Syntax:

Command `[SENSe:]STRain:APERture:ENABle <state> [,@<ch_list>]`

Query `[SENSe:]STRain:APERture:ENABle? [(@<ch_list>)]`

Parameters: `<state>` : 0 | 1 | OFF | ON

`<@ch_list>` : (@101)

Example:

Command `STR:APER:ENAB ON,(@101)`

18.9.3 [SENSe:]STRain:EXCitation

Description: Sets the excitation voltage value for the strain bridge.

Syntax:

Command `[SENSe:]STRain:EXCitation <voltage> [,@<ch_list>]`

Query `[SENSe:]STRain:EXCitation? [(@<ch_list>)]`

Parameters: `<voltage>` : <value> (Excitation Voltage)

`<@ch_list>` : (@101)

Example:

Command `STR:EXC 5,(@101)`

18.9.4 [SENSe:]STRain:EXCitation:TYPE

Description: Sets the excitation type (Internal or External).

Syntax:

Command `[SENSe:]STRain:EXCitation:TYPE <type> [,@<ch_list>]`

Query `[SENSe:]STRain:EXCitation:TYPE? [(@<ch_list>)]`

Parameters: `<type>` : INTernal | EXTernal

`<@ch_list>` : (@101)

Example:

Command `STR:EXC:TYPE EXT,(@101)`

18.9.5 [SENSe:]STRain:GFACtor

Description: Sets the Gauge Factor for the strain gauge.

Syntax:
Command [SENSe:]STRain:GFACtor <value> [,@<ch_list>]
Query [SENSe:]STRain:GFACtor? [(@<ch_list>)]
Parameters: <value> : <value> (Typical value 2.0)
<@ch_list> : (@101)
Example:
Command STR:GFAC 2.1,(@101)

18.9.6 [SENSe:]STRain:NPLCycles

Description: Sets the integration time in NPLC for strain measurements.

Syntax:
Command [SENSe:]STRain:NPLCycles <PLCs> [,@<ch_list>]
Query [SENSe:]STRain:NPLCycles? [(@<ch_list>)]
Parameters: <PLCs> : <value> (0.02 to 200)
<@ch_list> : (@101)
Example:
Command STR:NPLC 10,(@101)

18.9.7 [SENSe:]STRain:OCOMpensated

Description: Enables offset compensation for strain measurements.

Syntax:
Command [SENSe:]STRain:OCOMpensated <state> [,@<ch_list>]
Query [SENSe:]STRain:OCOMpensated? [(@<ch_list>)]
Parameters: <state> : OFF | ON
<@ch_list> : (@101)
Example:
Command STR:OCOM ON,(@101)

18.9.8 [SENSe:]STRain:POISson

Description: Sets the Poisson ratio for strain calculations.

Syntax:

Command [SENSe:]STRain:POISson <value> [,@<ch_list>]
Query [SENSe:]STRain:POISson? [(@<ch_list>)]
Parameters: <value> : <value> (Poisson Ratio e.g. 0.3)
<@ch_list> : (@101)
Example:
Command STR:POIS 0.3,(@101)

18.9.9 [SENSe:]STRain:RESistance

Description: Sets the nominal resistance of the strain gauge.

Syntax:
Command [SENSe:]STRain:RESistance <ohm> [,@<ch_list>]
Query [SENSe:]STRain:RESistance? [(@<ch_list>)]
Parameters: <ohm> : <value> (120, 350, etc.)
<@ch_list> : (@101)
Example:
Command STR:RES 120,(@101)

18.9.10 [SENSe:]STRain:UNSTrained

Description: Sets the unstrained reference value.

Syntax:
Command [SENSe:]STRain:UNSTrained <value> [,@<ch_list>]
Query [SENSe:]STRain:UNSTrained? [(@<ch_list>)]
Parameters: <value> : <value> (Reference value)
<@ch_list> : (@101)
Example:
Command STR:UNST 0,(@101)

18.9.11 [SENSe:]STRain:UNSTrained:IMMediate

Description: Measures the current value and sets it as the unstrained reference.

Syntax:
Command [SENSe:]STRain:UNSTrained:IMMediate [(@<ch_list>)]

Parameters: <@ch_list> : (@101)
Example:
Command STR:UNST:IMM (@101)

18.9.12 [SENSe:]STRain:VOLTage:RANGe

Description: Sets the voltage range for strain measurements.

Syntax:
Command [SENSe:]STRain:VOLTage:RANGe <range> [,@<ch_list>]
Query [SENSe:]STRain:VOLTage:RANGe? [(@<ch_list>)]
Parameters: <range> : <value> (100mV to 300V)
<@ch_list> : (@101)
Example:
Command STR:VOLT:RANG 10,(@101)

18.9.13 [SENSe:]STRain:VOLTage:RANGe:AUTO

Description: Enables auto-ranging for strain voltage measurements.

Syntax:
Command [SENSe:]STRain:VOLTage:RANGe:AUTO <state> [,@<ch_list>]
Query [SENSe:]STRain:VOLTage:RANGe:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)
Example:
Command STR:VOLT:RANG:AUTO ON,(@101)

18.9.14 [SENSe:]STRain:ZERO:AUTO

Description: Controls the auto-zero mode for strain measurements.

Syntax:
Command [SENSe:]STRain:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]STRain:ZERO:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)

Example:
Command `STR:ZERO:AUTO ON,(@101)`

18.10 SENSe TEMPerature Commands

18.10.1 [SENSe:]TEMPerature:APERture

Description: Sets the integration time in seconds for temperature measurements.

Syntax:
Command `[SENSe:]TEMPerature:APERture <seconds> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:APERture? [(@<ch_list>)]`
Parameters: `<seconds>` : <value> (Integration time)
`<@ch_list>` : (@101)
Example:
Command `TEMP:APER 0.1,(@101)`

18.10.2 [SENSe:]TEMPerature:APERture:ENABle

Description: Enables definition of integration time by aperture for temperature measurements.

Syntax:
Command `[SENSe:]TEMPerature:APERture:ENABle <state> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:APERture:ENABle? [(@<ch_list>)]`
Parameters: `<state>` : 0 | 1 | OFF | ON
`<@ch_list>` : (@101)
Example:
Command `TEMP:APER:ENAB ON,(@101)`

18.10.3 [SENSe:]TEMPerature:NPLCycles

Description: Sets the integration time in NPLC for temperature measurements.

Syntax:
Command `[SENSe:]TEMPerature:NPLCycles <PLCs> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:NPLCycles? [(@<ch_list>)]`

Parameters: <PLCs> : <value> (0.02 to 200)
<@ch_list> : (@101)
Example:
Command TEMP:NPLC 10,(@101)

18.10.4 [SENSe:]TEMPerature:RJUNction?

Description: Queries the internal reference junction temperature.

Syntax:
Query [SENSe:]TEMPerature:RJUNction? [(@<ch_list>)]
Parameters: <@ch_list> : (@101)
Example:
Query TEMP:RJUN? (@101)

18.10.5 [SENSe:]TEMPerature:RJUNction:SIMulated:AUTO:OFFSet

Description: Sets the offset for the simulated reference junction temperature.

Syntax:
Command [SENSe:]TEMPerature:RJUNction:SIMulated:AUTO:OFFSet <temp> [,@<ch_list>]
Query [SENSe:]TEMPerature:RJUNction:SIMulated:AUTO:OFFSet? [(@<ch_list>)]
Parameters: <temp> : <value> (Offset value)
<@ch_list> : (@101)
Example:
Command TEMP:RJUN:SIM:AUTO:OFFS 2.0,(@101)

18.10.6 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:TYPE

Description: Sets the RTD type.

Syntax:
Command [SENSe:]TEMPerature:TRANsducer:RTD:TYPE <type> [,@<ch_list>]
[SENSe:]TEMPerature:TRANsducer:FRTD:TYPE <type> [,@<ch_list>]
Query [SENSe:]TEMPerature:TRANsducer:RTD:TYPE? [(@<ch_list>)]
[SENSe:]TEMPerature:TRANsducer:FRTD:TYPE? [(@<ch_list>)]
Parameters: <type> : PT100 | D100 | F100 | PT385 | PT3916

<@ch_list> : (@101)

Example:

Command TEMP:TRAN:RTD:TYPE PT100,(@101)

18.10.7 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:USER:ALPHa|BETA|DELTa

Description: Sets the Alpha, Beta, or Delta coefficient for a User Defined RTD.

Syntax:

Command [SENSe:]TEMPerature:TRANsducer:RTD:USER:ALPHa <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:RTD:USER:BETA <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:RTD:USER:DELTa <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FRTD:USER:ALPHa <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FRTD:USER:BETA <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FRTD:USER:DELTa <value> [,@<ch_list>]

Query [SENSe:]TEMPerature:TRANsducer:RTD:USER:ALPHa? [(@<ch_list>)]

[SENSe:]TEMPerature:TRANsducer:RTD:USER:BETA? [(@<ch_list>)]

... (Queries match Set variations)

Parameters: <value> : <value> (Coefficient)

<@ch_list> : (@101)

Example:

Command TEMP:TRAN:RTD:USER:ALPH 0.00385,(@101)

18.10.8 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:OCOMpensated

Description: Enables offset compensation for RTD measurements.

Syntax:

Command [SENSe:]TEMPerature:TRANsducer:RTD:OCOMpensated <state> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FRTD:OCOMpensated <state> [,@<ch_list>]

Query [SENSe:]TEMPerature:TRANsducer:RTD:OCOMpensated? [(@<ch_list>)]

[SENSe:]TEMPerature:TRANsducer:FRTD:OCOMpensated? [(@<ch_list>)]

Parameters: <state> : OFF | ON

<@ch_list> : (@101)

Example:

Command TEMP:TRAN:RTD:OCOM ON,(@101)

18.10.9 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:POWer:LIMit[:STATe]

Description: Enables low-power mode for RTD measurements.

Syntax:

Command [SENSe:]TEMPerature:TRANsducer:RTD:POWer:LIMit[:STATe] <state> [,@<ch_list>

[SENSe:]TEMPerature:TRANsducer:FRTD:POWer:LIMit[:STATe] <state> [,@<ch_list

Query [SENSe:]TEMPerature:TRANsducer:RTD:POWer:LIMit[:STATe]? [(@<ch_list>)]

[SENSe:]TEMPerature:TRANsducer:FRTD:POWer:LIMit[:STATe]? [(@<ch_list>)]

Parameters: <state> : OFF | ON

<@ch_list> : (@101)

Example:

Command TEMP:TRAN:RTD:POW:LIM ON,(@101)

18.10.10 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:REFerence

Description: Sets the reference value (Ro) for the RTD.

Syntax:

Command [SENSe:]TEMPerature:TRANsducer:RTD:REFerence <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FRTD:REFerence <value> [,@<ch_list>]

Query [SENSe:]TEMPerature:TRANsducer:RTD:REFerence? [(@<ch_list>)]

[SENSe:]TEMPerature:TRANsducer:FRTD:REFerence? [(@<ch_list>)]

Parameters: <value> : <value> (Nominal resistance at 0C)

<@ch_list> : (@101)

Example:

Command TEMP:TRAN:RTD:REF 100,(@101)

18.10.11 [SENSe:]TEMPerature:TRANsducer:RTD|FRTD:RESistance[:REFerence]

Description: Alternative command to set the reference resistance (Ro).

Syntax:

Command `[SENSe:]TEMPerature:TRANsducer:RTD:RESistance[:REFerence] <value> [,@<ch_li`

`[SENSe:]TEMPerature:TRANsducer:FRTD:RESistance[:REFerence] <value>`
`[,@<ch_list>]`

Query `[SENSe:]TEMPerature:TRANsducer:RTD:RESistance[:REFerence]? [(@<ch_list>)]`

`[SENSe:]TEMPerature:TRANsducer:FRTD:RESistance[:REFerence]? [(@<ch_list>)]`

Parameters: `<value>` : <value> (Nominal resistance)
`<@ch_list>` : (@101)

Example:

Command `TEMP:TRAN:RTD:RES 100,(@101)`

18.10.12 [SENSe:]TEMPerature:TRANsducer:THER|FTH:POWer:LIMit[:STATe]

Description: Enables low-power mode for Thermistor measurements.

Syntax:

Command `[SENSe:]TEMPerature:TRANsducer:THERmistor:POWer:LIMit[:STATe] <state>`
`[,@<ch_list>]`
`[SENSe:]TEMPerature:TRANsducer:FTHermistor:POWer:LIMit[:STATe] <state>`
`[,@<ch_list>]`

Query `[SENSe:]TEMPerature:TRANsducer:THERmistor:POWer:LIMit[:STATe]? [(@<ch_list>`

`[SENSe:]TEMPerature:TRANsducer:FTHermistor:POWer:LIMit[:STATe]? [(@<ch_list`

Parameters: `<state>` : OFF | ON
`<@ch_list>` : (@101)

Example:

Command `TEMP:TRAN:THER:POW:LIM ON,(@101)`

18.10.13 [SENSe:]TEMPerature:TRANsducer:THER|FTH:REFerence

Description: Sets the nominal resistance for the thermistor.

Syntax:

Command [SENSe:]TEMPerature:TRANsducer:THERmistor:REFerence <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FTHermistor:REFerence <value> [,@<ch_list>]
Query [SENSe:]TEMPerature:TRANsducer:THERmistor:REFerence? [(@<ch_list>)]

[SENSe:]TEMPerature:TRANsducer:FTHermistor:REFerence? [(@<ch_list>)]

Parameters: <value> : <value> (Nominal resistance)
<@ch_list> : (@101)
Example:
Command TEMP:TRAN:THER:REF 2200,(@101)

18.10.14 [SENSe:]TEMPerature:TRANsducer:THER|FTH:TYPE

Description: Sets the thermistor type (e.g., 2.2k, 5k, 10k).

Syntax:
Command [SENSe:]TEMPerature:TRANsducer:THERmistor:TYPE <type> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FTHermistor:TYPE <type> [,@<ch_list>]
Query [SENSe:]TEMPerature:TRANsducer:THERmistor:TYPE? [(@<ch_list>)]
[SENSe:]TEMPerature:TRANsducer:FTHermistor:TYPE? [(@<ch_list>)]

Parameters: <type> : 2200 | 5000 | 10000 | USER
<@ch_list> : (@101)
Example:
Command TEMP:TRAN:THER:TYPE 5000,(@101)

18.10.15 [SENSe:]TEMPerature:TRANsducer:THER|FTH:USER:AVAL|BVAL|CVAL

Description: Sets the Steinhart-Hart coefficients (A, B, C) for a User Defined Thermistor.

Syntax:
Command [SENSe:]TEMPerature:TRANsducer:THERmistor:USER:AVALue <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:THERmistor:USER:BVALue <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:THERmistor:USER:CVALue <value> [,@<ch_list>]

[SENSe:]TEMPerature:TRANsducer:FTHermistor:USER:AVALue <value> [,@<ch_list>

```
          [SENSe:]TEMPerature:TRANsducer:FTHermistor:USER:BVALue <value> [,@<ch_list>

          [SENSe:]TEMPerature:TRANsducer:FTHermistor:USER:CVALue <value> [,@<ch_list>
Query     [SENSe:]TEMPerature:TRANsducer:THERmistor:USER:AVALue? [(@<ch_list>)]

          ... (Queries match Set variations)
```

Parameters: <value> : <value> (Coefficient)
<@ch_list> : (@101)
Example:
Command `TEMP:TRAN:THER:USER:AVAL 0.001,(@101)`

18.10.16 [SENSe:]TEMPerature:TRANsducer:TCouple:CHECk

Description: Enables the open thermocouple check.

Syntax:
Command `[SENSe:]TEMPerature:TRANsducer:TCouple:CHECk <state> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:TRANsducer:TCouple:CHECk? [(@<ch_list>)]`
Parameters: <state> : OFF | ON
<@ch_list> : (@101)
Example:
Command `TEMP:TRAN:TC:CHEC ON,(@101)`

18.10.17 [SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction

Description: Sets the external reference junction temperature when the RJunction type is set to EXTernal.

Syntax:
Command `[SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction <temp> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction? [(@<ch_list>)]`
Parameters: <temp> : <value> (Temperature in Celsius)
<@ch_list> : (@101)
Example:
Command `TEMP:TRAN:TC:RJUN 25.0,(@101)`

18.10.18 [SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction:TYPE

Description: Sets the reference junction type (Internal or External).

Syntax:

Command `[SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction:TYPE <type> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:TRANsducer:TCouple:RJUNction:TYPE? [(@<ch_list>)]`
Parameters: `<type>` : INTernal | EXTernal
`<@ch_list>` : (@101)
Example:
Command `TEMP:TRAN:TC:RJUN:TYPE INT,(@101)`

18.10.19 [SENSe:]TEMPerature:TRANsducer:TCouple:TYPE

Description: Sets the thermocouple type (B, E, J, K, N, R, S, T).

Syntax:

Command `[SENSe:]TEMPerature:TRANsducer:TCouple:TYPE <type> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:TRANsducer:TCouple:TYPE? [(@<ch_list>)]`
Parameters: `<type>` : B | E | J | K | N | R | S | T
`<@ch_list>` : (@101)
Example:
Command `TEMP:TRAN:TC:TYPE K,(@101)`

18.10.20 [SENSe:]TEMPerature:TRANsducer:TYPE

Description: Sets the temperature transducer type.

Syntax:

Command `[SENSe:]TEMPerature:TRANsducer:TYPE <type> [,@<ch_list>]`
Query `[SENSe:]TEMPerature:TRANsducer:TYPE? [(@<ch_list>)]`
Parameters: `<type>` : TC | RTD | FRTD | THER | FTH
`<@ch_list>` : (@101)
Example:
Command `TEMP:TRAN:TYPE TC,(@101)`

18.10.21 [SENSe:]TEMPerature:ZERO:AUTO

Description: Controls the auto-zero mode for temperature measurements.

Syntax:
Command [SENSe:]TEMPerature:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]TEMPerature:ZERO:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)
Example:
Command TEMP:ZERO:AUTO ON,(@101)

18.11 SENSe Totalizer Commands

18.11.1 [SENSe:]TOTalize:CLEar:IMMediate

Description: Immediately clears the count on the totalizer channels.

Syntax:
Command [SENSe:]TOTalize:CLEar:IMMediate [(@<ch_list>)]

Parameters: <@ch_list> : (@201)
Example:
Command TOT:CLE:IMM (@201)

18.11.2 [SENSe:]TOTalize:DATA?

Description: Reads the current count from the totalizer without resetting it.

Syntax:
Query [SENSe:]TOTalize:DATA? [(@<ch_list>)]
Parameters: <@ch_list> : (@201)
Example:
Query TOT:DATA? (@201)

18.11.3 [SENSe:]TOTalize:SLOPe

Description: Sets the edge polarity for counting (Positive rising edge or Negative falling edge).

Syntax:

Command [SENSe:]TOTalize:SLOPe <slope> [,@<ch_list>]
Query [SENSe:]TOTalize:SLOPe? [(@<ch_list>)]
Parameters: <slope> : POSitive | NEGative
<@ch_list> : (@201)
Example:
Command TOT:SLOP POS,(@201)

18.11.4 [SENSe:]TOTalize:STARt

Description: Starts the totalizer counting.

Syntax:
Command [SENSe:]TOTalize:STARt [(@<ch_list>)]

Parameters: <@ch_list> : (@201)
Example:
Command TOT:STAR (@201)

18.11.5 [SENSe:]TOTalize:STOP

Description: Stops the totalizer counting.

Syntax:
Command [SENSe:]TOTalize:STOP [(@<ch_list>)]

Parameters: <@ch_list> : (@201)
Example:
Command TOT:STOP (@201)

18.11.6 [SENSe:]TOTalize:TYPE

Description: Sets the totalizer read behavior. READ only reads the data; RRESet reads and then resets the count to zero.

Syntax:
Command [SENSe:]TOTalize:TYPE <type> [,@<ch_list>]
Query [SENSe:]TOTalize:TYPE? [(@<ch_list>)]
Parameters: <type> : READ | RRESet
<@ch_list> : (@201)

Example:
Command `TOT:TYPE RRES,(@201)`

18.12 SENSe VOLTage Commands

18.12.1 [SENSe:]VOLTage:AC:BANDwidth

Description: Sets the bandwidth filter for AC voltage measurements.

Syntax:
Command `[SENSe:]VOLTage:AC:BANDwidth <bandwidth> [,@<ch_list>]`
Query `[SENSe:]VOLTage:AC:BANDwidth? [(@<ch_list>)]`
Parameters: `<bandwidth>` : 3 | 20 | 200 | MIN | MAX | DEF (Hz)
`<@ch_list>` : (@101)
Example:
Command `VOLT:AC:BAND 20,(@101)`

18.12.2 [SENSe:]VOLTage:AC|DC:RANGe

Description: Sets the fixed measurement range for AC or DC voltage.

Syntax:
Command `[SENSe:]VOLTage:AC:RANGe <range> [,@<ch_list>]`
`[SENSe:]VOLTage:DC:RANGe <range> [,@<ch_list>]`
Query `[SENSe:]VOLTage:AC:RANGe? [(@<ch_list>)]`
`[SENSe:]VOLTage:DC:RANGe? [(@<ch_list>)]`
Parameters: `<range>` : <value> | MIN | MAX | DEF (100mV to 300V)
`<@ch_list>` : (@101)
Example:
Command `VOLT:DC:RANG 10,(@101)`

18.12.3 [SENSe:]VOLTage:AC|DC:RANGe:AUTO

Description: Enables auto-ranging for voltage measurements.

Syntax:

Command [SENSe:]VOLTage:AC:RANGe:AUTO <state> [,@<ch_list>]
[SENSe:]VOLTage:DC:RANGe:AUTO <state> [,@<ch_list>]
Query [SENSe:]VOLTage:AC:RANGe:AUTO? [(@<ch_list>)]
[SENSe:]VOLTage:DC:RANGe:AUTO? [(@<ch_list>)]

Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)

Example:

Command VOLT:DC:RANG:AUTO ON,(@101)

18.12.4 [SENSe:]VOLTage[:DC]:APERture

Description: Sets the integration time in seconds for DC voltage measurements.

Syntax:

Command [SENSe:]VOLTage[:DC]:APERture <seconds> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:APERture? [(@<ch_list>)]

Parameters: <seconds> : <value> (Integration time)
<@ch_list> : (@101)

Example:

Command VOLT:APER 0.1,(@101)

18.12.5 [SENSe:]VOLTage[:DC]:APERture:ENABle

Description: Enables definition of integration time by aperture for DC voltage measurements.

Syntax:

Command [SENSe:]VOLTage[:DC]:APERture:ENABle <state> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:APERture:ENABle? [(@<ch_list>)]

Parameters: <state> : 0 | 1 | OFF | ON
<@ch_list> : (@101)

Example:

Command VOLT:APER:ENAB ON,(@101)

18.12.6 [SENSe:]VOLTage[:DC]:IMPedance:AUTO

Description: Enables high input impedance (>10GΩ) for low DC voltage ranges to reduce loading errors.

Syntax:
Command [SENSe:]VOLTage[:DC]:IMPedance:AUTO <state> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:IMPedance:AUTO? [(@<ch_list>)]
Parameters: <state> : OFF | ON
<@ch_list> : (@101)
Example:
Command VOLT:IMP:AUTO ON,(@101)

18.12.7 [SENSe:]VOLTage[:DC]:NPLCycles

Description: Sets the integration time in NPLC for DC voltage measurements.

Syntax:
Command [SENSe:]VOLTage[:DC]:NPLCycles <PLCs> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:NPLCycles? [(@<ch_list>)]
Parameters: <PLCs> : <value> (0.02 to 200)
<@ch_list> : (@101)
Example:
Command VOLT:NPLC 10,(@101)

18.12.8 [SENSe:]VOLTage[:DC]:REFerence

Description: Sets a reference voltage value that is subtracted from subsequent DC voltage measurements (Relative mode).

Syntax:
Command [SENSe:]VOLTage[:DC]:REFerence <voltage> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:REFerence? [(@<ch_list>)]
Parameters: <voltage> : <value> | MIN | MAX | DEF (Reference value)
<@ch_list> : (@101)
Example:
Command VOLT:REF 1.5,(@101)

18.12.9 [SENSe:]VOLTage[:DC]:ZERO:AUTO

Description: Controls the auto-zero mode for DC voltage measurements.

Syntax:

Command [SENSe:]VOLTage[:DC]:ZERO:AUTO <state> [,@<ch_list>]
Query [SENSe:]VOLTage[:DC]:ZERO:AUTO? [(@<ch_list>)]

Parameters: <state> : OFF | ON | ONCE
<@ch_list> : (@101)

Example:

Command VOLT:ZERO:AUTO ON,(@101)

Source Subsystem

The Source subsystem commands control the generation of signals from the instrument. This includes setting the output voltage or current levels for DAC channels and writing data to the Digital I/O ports.

19.1 SOURce:CURRent

Description: Sets or queries the output current level for the specified DAC channels. This command is only effective if the source mode is set to Current.

Syntax:

Command `SOURce:CURRent <current>|MIN|MAX|DEF [(@<ch_list>)]`

Query `SOURce:CURRent? [(@<ch_list>)]`

Parameters: `<current>` : Output current value `<@ch_list>` : (@201)

Example:

Command `SOUR:CURR 0.01,(@201)`

Query `SOUR:CURR? (@201)`

19.2 SOURce:DIGital:DATA[:BYTE]

Description: Writes an 8-bit byte to the digital I/O port. This controls the logic state of lines 0-7.

Syntax:

Command `SOURce:DIGital:DATA[:BYTE] <data> [(@<ch_list>)]`

Query `SOURce:DIGital:DATA[:BYTE]? [(@<ch_list>)]`

Parameters: `<data>` : 0 to 255 `<@ch_list>` : (@201)

Example:

Command `SOUR:DIG:DATA 255,(@201)`

Query `SOUR:DIG:DATA? (@201)`

19.3 SOURce:DIGital:DATA:WORD

Description: Writes a 16-bit word to the digital I/O port. This controls the logic state of lines 0-15 (if supported by the module).

Syntax:

Command `SOURce:DIGital:DATA:WORD <data> [(@<ch_list>)]`

Query `SOURce:DIGital:DATA:WORD? [(@<ch_list>)]`

Parameters: `<data>` : 0 to 65535 `<@ch_list>` : (@201)

Example:

Command `SOUR:DIG:DATA:WORD 65535,(@201)`

Query `SOUR:DIG:DATA:WORD? (@201)`

19.4 SOURce:MODE

Description: Sets or queries the output mode of the specified DAC channels (Voltage or Current).

Syntax:

Command `SOURce:MODE CURRent|VOLTage [(@<ch_list>)]`
Query `SOURce:MODE? [(@<ch_list>)]`

Parameters: `<mode>` : CURRent | VOLTage `<@ch_list>` : (@201)

Example:

Command `SOUR:MODE VOLT,(@201)`
Query `SOUR:MODE? (@201)`

19.5 SOURce:MODE:LOCK

Description: Locks the source mode to prevent accidental changes. When locked, the `SOURce:MODE` command cannot change the output type.

Syntax:

Command `SOURce:MODE:LOCK OFF|ON [(@<ch_list>)]`
Query `SOURce:MODE:LOCK? [(@<ch_list>)]`

Parameters: `<state>` : OFF | ON `<@ch_list>` : (@201)

Example:

Command `SOUR:MODE:LOCK ON,(@201)`
Query `SOUR:MODE:LOCK? (@201)`

19.6 SOURce:VOLTage

Description: Sets or queries the output voltage level for the specified DAC channels. This command is only effective if the source mode is set to Voltage.

Syntax:

Command `SOURce:VOLTage <voltage>|MIN|MAX|DEF [(@<ch_list>)]`
Query `SOURce:VOLTage? [(@<ch_list>)]`

Parameters: `<voltage>` : Output voltage value `<@ch_list>` : (@201)

Example:

Command `SOUR:VOLT 5.0,(@201)`
Query `SOUR:VOLT? (@201)`

Status Subsystem

The Status subsystem controls the instrument's status registers. These registers allow the user to monitor the state of the instrument, detect errors, and verify the completion of operations.
The status system consists of three main register groups:

- **Operation Status Register:** Monitors normal instrument operations.
- **Questionable Status Register:** Monitors errors and questionable data conditions.
- **Alarm Status Register:** Monitors limit violations on measurement channels.

20.1 STATus:ALARm:CONDition?

Description: Queries the Condition register for the Alarm group. This register shows the *real-time* status of the alarms. If a bit is set, the corresponding alarm is currently active.

Syntax:
Query `STATus:ALARm:CONDition?`
Parameters: None
Example:
Query `STAT:ALAR:COND?`

NOTICE **Bits 0-3 correspond to Alarm Output lines 1-4. Bits 4-20 correspond to specific channels depending on configuration.**

20.2 STATus:ALARm:ENABle

Description: Sets or queries the Enable register for the Alarm group. Setting a bit in this register allows the corresponding event to be reported to the Status Byte.

Syntax:
Command `STATus:ALARm:ENABle <value>`
Query `STATus:ALARm:ENABle?`
Parameters: `<value>` : 0 to 65535
Example:
Command `STAT:ALAR:ENAB 15`
Query `STAT:ALAR:ENAB?`

20.3 STATus:ALARm[:EVENt]?

Description: Queries the Event register for the Alarm group. This is a latched register; once a bit is set, it remains set until cleared by reading this register or sending `*CLS`.

Syntax:
Query `STATus:ALARm[:EVENt]?`
Parameters: None
Example:
Query `STAT:ALAR?`

20.4 STATus:OPERation:CONDition?

Description: Queries the Condition register for the Standard Operation group. Returns the real-time state of the instrument's operations (e.g., Calibrating, Measuring, Waiting for Trigger).

Syntax:
Query `STATus:OPERation:CONDition?`
Parameters: None
Example:
Query `STAT:OPER:COND?`

20.5 STATus:OPERation:ENABle

Description: Sets or queries the Enable register for the Standard Operation group. Bits set here enable the corresponding events to generate a summary bit in the Status Byte.

Syntax:
Command `STATus:OPERation:ENABle <value>`
Query `STATus:OPERation:ENABle?`
Parameters: `<value>` : 0 to 65535
Example:
Command `STAT:OPER:ENAB 16`
Query `STAT:OPER:ENAB?`

20.6 STATus:OPERation[:EVENt]?

Description: Queries and clears the Event register for the Standard Operation group. Returns a decimal value representing the bits set.

Syntax:
Query `STATus:OPERation[:EVENt]?`
Parameters: None
Example:
Query `STAT:OPER?`

20.7 STATus:PRESet

Description: Clears all Enable registers and sets them to their power-on default values. It does not affect the Event or Condition registers.

Syntax:
Command `STATus:PRESet`

Parameters: None

Example:
Command `STAT:PRES`

20.8 STATus:QUEStionable:CONDition?

Description: Queries the Condition register for the Questionable Status group. This indicates if a problem currently exists (e.g., Voltage Overload, Over Temperature).

Syntax:
Query `STATus:QUEStionable:CONDition?`

Parameters: None

Example:
Query `STAT:QUES:COND?`

20.9 STATus:QUEStionable:ENABle

Description: Sets or queries the Enable register for the Questionable Status group.

Syntax:
Command `STATus:QUEStionable:ENABle <value>`
Query `STATus:QUEStionable:ENABle?`

Parameters: `<value>` : 0 to 65535

Example:
Command `STAT:QUES:ENAB 256`
Query `STAT:QUES:ENAB?`

20.10 STATus:QUEStionable[:EVENt]?

Description: Queries and clears the Event register for the Questionable Status group. Reading this register clears it.

Syntax:
Query `STATus:QUEStionable[:EVENt]?`

Parameters: None

Example:

Query `STAT:QUES?`

System Subsystem

The System subsystem commands enable you to query the instrument's error queue, control hardware settings like the beeper and relay cycle counters, and configure the remote communication interfaces (GPIB and LAN).

21.1 SYSTem Related Commands

21.1.1 SYSTem:ALARm?

Description: Queries the alarm status of the instrument. Returns a value indicating if any alarm is currently asserted.

Syntax:
Query `SYSTem:ALARm?`
Parameters: None
Example:
Query `SYST:ALAR?`

21.1.2 SYSTem:BEEPer[:IMMediate]

Description: Issues a single beep immediately.

Syntax:
Command `SYSTem:BEEPer[:IMMediate]`
Query `SYSTem:BEEPer[:IMMediate]?`
Parameters: None
Example:
Command `SYST:BEEP`

21.1.3 SYSTem:BEEPer:ERRor

Description: Enables or disables the beep when an error occurs in the error queue.

Syntax:
Command `SYSTem:BEEPer:ERRor OFF|ON`
Query `SYSTem:BEEPer:ERRor?`
Parameters: `<state>` : OFF | ON
Example:
Command `SYST:BEEP:ERR ON`
Query `SYST:BEEP:ERR?`

21.1.4 SYSTem:BEEPer:STATe

Description: Enables or disables the system beeper for all functions.

Syntax:

Command `SYSTem:BEEPer:STATe OFF|ON`

Query `SYSTem:BEEPer:STATe?`

Parameters: `<state>` : OFF | ON

Example:

Command `SYST:BEEP:STAT OFF`

Query `SYST:BEEP:STAT?`

21.1.5 SYSTem:CLICk:STATe

Description: Enables or disables the key click sound when front panel keys are pressed.

Syntax:

Command `SYSTem:CLICk:STATe OFF|ON`

Query `SYSTem:CLICk:STATe?`

Parameters: `<state>` : OFF | ON

Example:

Command `SYST:CLIC:STAT OFF`

Query `SYST:CLIC:STAT?`

21.1.6 SYSTem:CPON

Description: Card Power On. Reinitializes the plug-in modules, effectively cycling power to them.

Syntax:

Command `SYSTem:CPON`

Parameters: None

Example:

Command `SYST:CPON`

21.1.7 SYSTem:CTYPe?

Description: Queries the type of module installed in the specified slot.

Syntax:
Query SYSTem:CTYPe? <slot>
Parameters: <slot> : 1 to 3
Example:
Query SYST:CTYP? 1

21.1.8 SYSTem:DATE

Description: Sets or queries the system date.

Syntax:
Command SYSTem:DATE <year>,<month>,<day>
Query SYSTem:DATE?
Parameters: <year> : 2000 to 2099 <month> : 1 to 12 <day> : 1 to 31
Example:
Command SYST:DATE 2025,1,15
Query SYST:DATE?

21.1.9 SYSTem:ERRor?

Description: Queries and clears the next error from the error queue. Returns error code and message string.

Syntax:
Query SYSTem:ERRor?
Parameters: None
Example:
Query SYST:ERR?

21.1.10 SYSTem:LFRequency?

Description: Queries the power line frequency detected by the instrument (50 Hz or 60 Hz).

Syntax:
Query SYSTem:LFRequency?
Parameters: None
Example:

Query SYST:LFR?

21.1.11 SYSTem:LOCal

Description: Places the instrument in Local mode, enabling the front panel keys. Used to unlock the front panel after remote operation.

Syntax:

Command SYSTem:LOCal

Parameters: None

Example:

Command SYST:LOC

21.1.12 SYSTem:REMote

Description: Places the instrument in Remote mode. Front panel keys are locked except for the Local key.

Syntax:

Command SYSTem:REMote

Parameters: None

Example:

Command SYST:REM

21.1.13 SYSTem:PARameter:LOAD

Description: Loads the instrument state parameters from a specified memory file.

Syntax:

Command SYSTem:PARameter:LOAD <filename>

Parameters: <filename> : Quoted string

Example:

Command SYST:PAR:LOAD "Config1"

21.1.14 SYSTem:PARameter:SAVE

Description: Saves the current instrument state parameters to a file.

Syntax:

Command `SYSTem:PARameter:SAVE <filename>`

Parameters: `<filename>` : Quoted string

Example:

Command `SYST:PAR:SAVE "Config1"`

21.1.15 SYSTem:PERSona[:MANufacturer]

Description: Sets the manufacturer string returned by *IDN? to emulate legacy equipment.

Syntax:

Command `SYSTem:PERSona[:MANufacturer] <string>`
Query `SYSTem:PERSona[:MANufacturer]?`

Parameters: `<string>` : "Heuristic", etc.

Example:

Command `SYST:PERS "Heuristic"`
Query `SYST:PERS?`

21.1.16 SYSTem:PERSona[:MANufacturer]:DEFault

Description: Resets the manufacturer string to the factory default.

Syntax:

Command `SYSTem:PERSona[:MANufacturer]:DEFault`

Parameters: None

Example:

Command `SYST:PERS:DEF`

21.1.17 SYSTem:PERSona:MODel

Description: Sets the model string returned by *IDN?.

Syntax:

Command SYSTem:PERSona:MODel <string>
Query SYSTem:PERSona:MODel?
Parameters: <string> : "DAQ9600", etc.
Example:
Command SYST:PERS:MOD "DAQ9600"
Query SYST:PERS:MOD?

21.1.18 SYSTem:PERSona:MODel:DEFault

Description: Resets the model string to the factory default.

Syntax:
Command SYSTem:PERSona:MODel:DEFault

Parameters: None
Example:
Command SYST:PERS:MOD:DEF

21.1.19 SYSTem:PRESet

Description: Resets the instrument to a known state. Similar to *RST but may configure specific preferences.

Syntax:
Command SYSTem:PRESet

Parameters: None
Example:
Command SYST:PRES

21.1.20 SYSTem:RELay:CYCLes?

Description: Queries the cycle count for the relays on the specified channels. Useful for preventative maintenance.

Syntax:
Query SYSTem:RELay:CYCLes? (@<ch_list>)
Parameters: <@ch_list> : (@101)
Example:

Query SYST:REL:CYCL? (@101)

21.1.21 SYSTem:RELay:CYCLes:CLEar

Description: Clears the cycle count for the specified relays. Typically done after relay replacement.

Syntax:

Command SYSTem:RELay:CYCLes:CLEar (@<ch_list>)

Parameters: <@ch_list> : (@101)

Example:

Command SYST:REL:CYCL:CLE (@101)

21.1.22 SYSTem:RELay:CYCLes:FACTory?

Description: Queries the factory cycle count (total lifetime cycles) for the relays.

Syntax:

Query SYSTem:RELay:CYCLes:FACTory? (@<ch_list>)

Parameters: <@ch_list> : (@101)

Example:

Query SYST:REL:CYCL:FACT? (@101)

21.1.23 SYSTem:SCPi:MODE

Description: Sets the SCPI compatibility mode (e.g., Native or Compatible with other vendors).

Syntax:

Command SYSTem:SCPi:MODE NATive|COMPatible

Query SYSTem:SCPi:MODE?

Parameters: <mode> : NATive | COMPatible

Example:

Command SYST:SCP:MODE NAT

Query SYST:SCP:MODE?

21.1.24 SYSTem:SCPi:AUTO:SAVE

Description: Enables automatic saving of the SCPI configuration.

Syntax:
Command `SYSTem:SCPi:AUTO:SAVE OFF|ON`
Query `SYSTem:SCPi:AUTO:SAVE?`
Parameters: `<state>` : OFF | ON
Example:
Command `SYST:SCP:AUTO:SAVE ON`
Query `SYST:SCP:AUTO:SAVE?`

21.1.25 SYSTem:SERial?

Description: Queries the instrument's serial number.

Syntax:
Query `SYSTem:SERial?`
Parameters: None
Example:
Query `SYST:SER?`

21.1.26 SYSTem:SLOT:LABel

Description: Assigns a user-defined label string to a specific slot.

Syntax:
Command `SYSTem:SLOT:LABel <slot>,<string>`
Query `SYSTem:SLOT:LABel? <slot>`
Parameters: `<slot>` : 1 to 3 `<string>` : "Temp_Module"
Example:
Command `SYST:SLOT:LAB 1,"Module1"`
Query `SYST:SLOT:LAB? 1`

21.1.27 SYSTem:TEMPerature?

Description: Queries the internal temperature of the instrument.

Syntax:
Query `SYSTem:TEMPerature?`
Parameters: None

Example:
Query SYST:TEMP?

21.1.28 SYSTem:TIME

Description: Sets or queries the system time.

Syntax:
Command SYSTem:TIME <hour>,<minute>,<second>
Query SYSTem:TIME?
Parameters: <hour> : 0 to 23 <minute> : 0 to 59 <second> : 0 to 59
Example:
Command SYST:TIME 14,30,00
Query SYST:TIME?

21.1.29 SYSTem:TIME:SCAN?

Description: Queries the scan start time of the last scan operation.

Syntax:
Query SYSTem:TIME:SCAN?
Parameters: None
Example:
Query SYST:TIME:SCAN?

21.1.30 SYSTem:UPTime?

Description: Queries the time elapsed since the instrument was powered on.

Syntax:
Query SYSTem:UPTime?
Parameters: None
Example:
Query SYST:UPT?

21.1.31 SYSTem:VERSion?

Description: Queries the SCPI version that the instrument complies with.

Syntax:
Query `SYSTem:VERSion?`
Parameters: None
Example:
Query `SYST:VERS?`

21.1.32 SYSTem:WMESsage

Description: Sets or queries the wake-up message displayed on power-on.

Syntax:
Command `SYSTem:WMESsage <string>`
Query `SYSTem:WMESsage?`
Parameters: `<string>` : Quoted string (max 40 chars)
Example:
Command `SYST:WMES "Lab 1"`
Query `SYST:WMES?`

21.2 SYSTem COMMunication Commands

21.2.1 SYSTem:COMMunicate:GPIB:ADDRess

Description: Sets or queries the GPIB address of the instrument.

Syntax:
Command `SYSTem:COMMunicate:GPIB:ADDRess <value>`
Query `SYSTem:COMMunicate:GPIB:ADDRess?`
Parameters: `<value>` : 0 to 30
Example:
Command `SYST:COMM:GPIB:ADDR 5`
Query `SYST:COMM:GPIB:ADDR?`

21.2.2 SYSTem:COMMunicate:LAN:DHCP

Description: Enables or disables DHCP. When ON, the IP address is assigned automatically.

Syntax:

Command `SYSTem:COMMunicate:LAN:DHCP OFF|ON`

Query `SYSTem:COMMunicate:LAN:DHCP?`

Parameters: `<state>` : OFF | ON

Example:

Command `SYST:COMM:LAN:DHCP ON`

Query `SYST:COMM:LAN:DHCP?`

21.2.3 SYSTem:COMMunicate:LAN:DNS

Description: Sets or queries the DNS server addresses. [X] represents the index (1 or 2).

Syntax:

Command `SYSTem:COMMunicate:LAN:DNS[X] "<address>"`

Query `SYSTem:COMMunicate:LAN:DNS[X]?`

Parameters: `<address>` : "192.168.1.1"

Example:

Command `SYST:COMM:LAN:DNS1 "8.8.8.8"`

Query `SYST:COMM:LAN:DNS1?`

21.2.4 SYSTem:COMMunicate:LAN:DOMain?

Description: Queries the domain name of the network.

Syntax:

Query `SYSTem:COMMunicate:LAN:DOMain?`

Parameters: None

Example:

Query `SYST:COMM:LAN:DOM?`

21.2.5 SYSTem:COMMunicate:LAN:GATeway

Description: Sets or queries the default gateway IP address.

Syntax:
Command `SYSTem:COMMunicate:LAN:GATeway "<address>"`
Query `SYSTem:COMMunicate:LAN:GATeway?`
Parameters: `<address>` : "192.168.1.254"
Example:
Command `SYST:COMM:LAN:GAT "192.168.1.1"`
Query `SYST:COMM:LAN:GAT?`

21.2.6 SYSTem:COMMunicate:LAN:HOSTname

Description: Sets or queries the instrument's hostname.

Syntax:
Command `SYSTem:COMMunicate:LAN:HOSTname "<name>"`
Query `SYSTem:COMMunicate:LAN:HOSTname?`
Parameters: `<name>` : String
Example:
Command `SYST:COMM:LAN:HOST "DAQ-LAB"`
Query `SYST:COMM:LAN:HOST?`

21.2.7 SYSTem:COMMunicate:LAN:IPADdress

Description: Sets or queries the static IP address. Only effective if DHCP is OFF.

Syntax:
Command `SYSTem:COMMunicate:LAN:IPADdress "<address>"`
Query `SYSTem:COMMunicate:LAN:IPADdress?`
Parameters: `<address>` : "192.168.1.100"
Example:
Command `SYST:COMM:LAN:IPAD "10.0.0.5"`
Query `SYST:COMM:LAN:IPAD?`

21.2.8 SYSTem:COMMunicate:LAN:MAC?

Description: Queries the MAC address of the LAN interface.

Syntax:

Query SYSTem:COMMunicate:LAN:MAC?
Parameters: None
Example:
Query SYST:COMM:LAN:MAC?

21.2.9 SYSTem:COMMunicate:LAN:SMASk

Description: Sets or queries the subnet mask.

Syntax:
Command SYSTem:COMMunicate:LAN:SMASk "<mask>"
Query SYSTem:COMMunicate:LAN:SMASk?
Parameters: <mask> : "255.255.255.0"
Example:
Command SYST:COMM:LAN:SMAS "255.255.0.0"
Query SYST:COMM:LAN:SMAS?

21.2.10 SYSTem:COMMunicate:LAN:TCP:ENABle

Description: Enables or disables the TCP/IP socket service.

Syntax:
Command SYSTem:COMMunicate:LAN:TCP:ENABle OFF|ON
Query SYSTem:COMMunicate:LAN:TCP:ENABle?
Parameters: <state> : OFF | ON
Example:
Command SYST:COMM:LAN:TCP:ENAB ON
Query SYST:COMM:LAN:TCP:ENAB?

21.2.11 SYSTem:COMMunicate:LAN:TCP:PORT

Description: Sets or queries the port number for the TCP socket service.

Syntax:
Command SYSTem:COMMunicate:LAN:TCP:PORT <port>
Query SYSTem:COMMunicate:LAN:TCP:PORT?
Parameters: <port> : Default 5025

Example:
Command SYST:COMM:LAN:TCP:PORT 5025
Query SYST:COMM:LAN:TCP:PORT?

21.2.12 SYSTem:COMMunicate:LAN:TELNet:ECHO

Description: Enables or disables echo for the Telnet session.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:ECHO OFF|ON
Query SYSTem:COMMunicate:LAN:TELNet:ECHO?
Parameters: <state> : OFF | ON
Example:
Command SYST:COMM:LAN:TELN:ECHO ON
Query SYST:COMM:LAN:TELN:ECHO?

21.2.13 SYSTem:COMMunicate:LAN:TELNet:ENABle

Description: Enables or disables the Telnet service.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:ENABle OFF|ON
Query SYSTem:COMMunicate:LAN:TELNet:ENABle?
Parameters: <state> : OFF | ON
Example:
Command SYST:COMM:LAN:TELN:ENAB ON
Query SYST:COMM:LAN:TELN:ENAB?

21.2.14 SYSTem:COMMunicate:LAN:TELNet:TIMeout

Description: Sets the inactivity timeout for the Telnet session in seconds.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:TIMeout <seconds>
Query SYSTem:COMMunicate:LAN:TELNet:TIMeout?
Parameters: <seconds> : 0 to 3600
Example:

Command SYST:COMM:LAN:TELN:TIM 300
Query SYST:COMM:LAN:TELN:TIM?

21.2.15 SYSTem:COMMunicate:LAN:TELNet:PORT

Description: Sets the port number for the Telnet service.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:PORT <port>
Query SYSTem:COMMunicate:LAN:TELNet:PORT?
Parameters: <port> : Default 5024
Example:
Command SYST:COMM:LAN:TELN:PORT 5024
Query SYST:COMM:LAN:TELN:PORT?

21.2.16 SYSTem:COMMunicate:LAN:TELNet:PROMpt

Description: Sets the command prompt string displayed in the Telnet session.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:PROMpt "<string>"
Query SYSTem:COMMunicate:LAN:TELNet:PROMpt?
Parameters: <string> : "DAQ>"
Example:
Command SYST:COMM:LAN:TELN:PROM "SCPI>"
Query SYST:COMM:LAN:TELN:PROM?

21.2.17 SYSTem:COMMunicate:LAN:TELNet:WMESsage

Description: Sets the welcome message displayed when connecting via Telnet.

Syntax:
Command SYSTem:COMMunicate:LAN:TELNet:WMESsage "<string>"
Query SYSTem:COMMunicate:LAN:TELNet:WMESsage?
Parameters: <string> : Welcome message
Example:
Command SYST:COMM:LAN:TELN:WMES "Welcome"
Query SYST:COMM:LAN:TELN:WMES?

21.2.18 SYSTem:COMMunicate:LAN:TIMeout

Description: Sets the general LAN inactivity timeout.

Syntax:

Command `SYSTem:COMMunicate:LAN:TIMeout <seconds>`

Query `SYSTem:COMMunicate:LAN:TIMeout?`

Parameters: `<seconds>` : Time in seconds

Example:

Command `SYST:COMM:LAN:TIM 60`

Query `SYST:COMM:LAN:TIM?`

21.2.19 SYSTem:COMMunicate:LAN:UPDate

Description: Applies changes made to LAN settings. Network changes may not take effect until this command is sent.

Syntax:

Command `SYSTem:COMMunicate:LAN:UPDate`

Parameters: None

Example:

Command `SYST:COMM:LAN:UPD`

21.2.20 SYSTem:COMMunicate:LAN:WEB:ENABle

Description: Enables or disables the built-in Web Server.

Syntax:

Command `SYSTem:COMMunicate:LAN:WEB:ENABle OFF|ON`

Query `SYSTem:COMMunicate:LAN:WEB:ENABle?`

Parameters: `<state>` : OFF | ON

Example:

Command `SYST:COMM:LAN:WEB:ENAB ON`

Query `SYST:COMM:LAN:WEB:ENAB?`

21.2.21 SYSTem:COMMunicate:LAN:WINS

Description: Sets or queries the WINS server address.

Syntax:

Command `SYSTem:COMMunicate:LAN:WINS "<address>"`

Query `SYSTem:COMMunicate:LAN:WINS?`

Parameters: `<address>` : IP Address

Example:

Command `SYST:COMM:LAN:WINS "192.168.1.50"`

Query `SYST:COMM:LAN:WINS?`

Trigger Subsystem

The Trigger subsystem commands control the triggering behavior of the instrument. These commands determine how and when a scan is initiated, the number of times a scan is repeated (Trigger Count), and the specific source of the trigger event (Internal Timer, External signal, Bus command, or Immediate).

22.1 TRIGger:COUNt

Description: Sets or queries the number of triggers the instrument will accept before returning to the "idle" trigger state. This effectively determines the number of times the scan list is swept per `INITiate` command.

Syntax:

Command `TRIGger:COUNt <count>|MIN|MAX|DEF|INFinity`

Query `TRIGger:COUNt?`

Parameters: `<count>` : 1 to 1,000,000 or INFinity

Example:

Command `TRIG:COUN 10`

Query `TRIG:COUN?`

NOTICE **If set to INFinity, the instrument continues scanning until an `ABORt` command is sent.**

22.2 TRIGger:SLOPe

Description: Sets or queries the edge polarity for the external trigger signal input on the rear panel. This command is only relevant when `TRIGger:SOURce` is set to EXTernal.

Syntax:

Command `TRIGger:SLOPe POSitive|NEGative`

Query `TRIGger:SLOPe?`

Parameters: `<slope>` : POSitive (Rising Edge) | NEGative (Falling Edge)

Example:

Command `TRIG:SLOP NEG`

Query `TRIG:SLOP?`

22.3 TRIGger:SOURce

Description: Sets or queries the trigger source.

- **IMMediate:** The scan starts immediately after `INITiate` is sent.
- **TIMer:** The instrument scans at intervals defined by `TRIGger:TIMer`.
- **EXTernal:** The instrument waits for a TTL pulse on the rear-panel Ext Trig input.

- **BUS:** The instrument waits for a software trigger (*TRG).
- **ALARm1..4:** The instrument triggers on a specific alarm event.

Syntax:
Command `TRIGger:SOURce IMMediate|TIMer|EXTernal|BUS|ALARm1|ALARm2|ALARm3|ALARm4`
Query `TRIGger:SOURce?`
Parameters: `<source>` : IMM | TIM | EXT | BUS | ALARm
Example:
Command `TRIG:SOUR TIM`
Query `TRIG:SOUR?`

22.4 TRIGger:TIMer

Description: Sets or queries the time interval between scan sweeps when `TRIGger:SOURce` is set to TIMer.

Syntax:
Command `TRIGger:TIMer <seconds>|MIN|MAX|DEF`
Query `TRIGger:TIMer?`
Parameters: `<seconds>` : 0 to 360000 seconds (resolution 1ms)
Example:
Command `TRIG:TIM 5`
Query `TRIG:TIM?`
