Agilent B2900 Series
Precision Source/Measure Unit

SCPI Command Reference

Agilent Technologies

Notices

Manual Part Number

B2910-90030

Edition

Edition 1, March 2011
Edition 2, July 2011

Agilent Technologies, Inc.
5301 Stevens Creek Blvd
Santa Clara, CA 95051 USA

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Open Software License

A portion of the software in this product is licensed under terms of the General Public License Version 2 ("GPLv2"). The text of the license and source code can be found at:

www.agilent.com/find/GPLV2

About Customer Feedback

We love hearing from you. Please take a few moments and let us know how we can improve this manual. Click here to open the Agilent B2900 manual feedback form.

We respect your privacy. Be assured that Agilent will never sell or rent your information. Nor will Agilent share this information with other companies without your expressed consent. We make a commitment to you that we will respect and protect your privacy. Please see the details of this commitment in our Privacy Statement. Click here to open the statement.

In This Manual

This manual contains reference information to help you program the Agilent B2900 Source/Measure Unit series over the remote interface using the SCPI programming language. The Agilent B2900 supports the SCPI programming language on all of its remote I/O interfaces.

- Chapter 1, “Programming Basics.”

  Describes a basic information for programming the B2900, and contains a brief introduction to the SCPI programming language, the data output format, the status system diagram, and the non-volatile settings.

- Chapter 2, “Subsystem Command Summary.”

  Lists the B2900 SCPI subsystem commands and summary descriptions.

- Chapter 3, “Common Commands.”

  Provides reference information such as description and command syntax of SCPI common commands available for the B2900.

- Chapter 4, “Subsystem Commands.”

  Provides reference information such as description and command syntax of device specific SCPI commands available for the B2900.

- Chapter 5, “Error Messages.”

  Lists the B2900 error messages, and provides error number, error message and description.

- Chapter 6, “Using Your Existing Programs.”

  Describes how to use the existing programs which you created for controlling conventional source/measure instruments.

See *User's Guide* for information about the B2900 itself.

Refer to *Programming Guide* to create a B2900 control program.

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

1 Programming Basics

This chapter describes a basic information for programming Agilent B2900, and consists of the following sections.

- “SCPI Commands”
- “SCPI Messages”
- “SCPI Command Completion”
- “SCPI Conventions and Data Formats”
- “Data Output Format”
- “GPIB Capability”
- “Status Byte”
- “Status System Diagram”
- “Non-Volatile Settings”

SCPI Commands

SCPI (Standard Commands for Programmable Instruments) is a programming language for controlling test and measurement instruments. SCPI provides instrument control with a standardized command syntax and style, as well as a standardized data interchange format.

SCPI has two types of commands, common and subsystem.

- Common commands

  Common commands are defined by the IEEE 488.2 standard to perform common interface functions such as reset, status, and synchronization. All common commands consist of a three-letter mnemonic preceded by an asterisk: *RST, *IDN?, *SRE 8. Common commands belong to the IEEE-488.2 Common Commands group.

- Subsystem commands

  Subsystem commands perform specific instrument functions. They can be a single command or a group of commands. The groups are comprised of commands that extend one or more levels below the root. Subsystem commands are arranged alphabetically according to the function they perform. The following example shows a portion of a subsystem command tree, from which you access the commands located along the various paths. Some [optional] commands have been included for clarity.

  Example:

```
:OUTPut
      :SAVE <index>
     [:STATe] <Bool>

[:SOURce]
       :FUNCtion
            :MODE <mode>
           [:SHAPe] <shape>
       :VOLTage
            :MODE <mode>
            :RANGe <range>
                  :AUTO <Bool>
```

Multiple Commands in a Message

Multiple SCPI commands can be combined and sent as a single message with one message terminator. There are two important considerations when sending several commands within a single message.

- Use a semicolon to separate commands within a message.
- There is an implied header path that affects how commands are interpreted by the instrument.

The header path can be thought of as a string that is inserted before each command within a message. For the first command in a message, the header path is a null string. For each subsequent command, the header path is defined as the characters that make up the headers of the previous command in the message up to and including the last colon separator. An example of a message with two commands is:

```
OUTPut:STATe ON;PROTection ON
```

which shows the use of the semicolon separating the two commands, and also illustrates the header path concept. Note that with the second command, the leading header “OUTPut” was omitted because after the “OUTPut:STATe ON” command, the header path became defined as “OUTPut” and thus the instrument interpreted the second command as:

```
OUTPut:PROTection ON
```

In fact, it would have been syntactically incorrect to include the “OUTPut” explicitly in the second command, since the result after combining it with the header path would be:

```
OUTPut:OUTPut:PROTection ON
```

which is incorrect.

Moving Between Subsystems

In order to combine commands from different subsystems, you must reset the header path to a null string within a message. This is done by beginning the command with a colon (:), which discards any previous header path. For example, you could disable the output relay protection function and check the status of the Operation Condition register with a single message by using a root specifier as follows:

```
OUTPut:PROTection OFF;:STATus:OPERation:CONDition?
```

The following message shows how to combine commands from different subsystems as well as within the same subsystem:

```
VOLTage:LEVel 7.5;RANGe 10;:CURRent:LEVel 0.1
```

Note the use of the optional header LEVel to maintain the correct path within the subsystems, and the use of the root specifier to move between subsystems.

Including Common Commands

You can combine common commands and subsystem commands into a single message. Treat the common command as a message unit by separating it with a semicolon (the message unit separator). Common commands do not affect the header path; you may insert them anywhere in the message.

```
OUTPut OFF;*RCL 1;OUTPut ON
```

Using Queries

Observe the following precautions when using queries.

- Add a blank space between the query indicator (?) and any subsequent parameter such as a channel list.
- Allocate a proper number of variables for the returned data.
- Read back all the results of a query before sending another command to the instrument. Otherwise, a Query Interrupted error will occur and the unreturned data will be lost.

Coupled Commands

When commands are coupled, it means that the value sent by one command is affected by the settings of another command. The following commands are coupled:

`[SOURce:]CURRent` and `[SOURce:]CURRent:RANGe`

`[SOURce:]VOLTage` and `[SOURce:]VOLTage:RANGe`

If a range command is sent to place an output into a range with a lower maximum setting than the present level, an error is generated. This also occurs if a level is programmed with a value too large for the present range.

These types of errors can be avoided by sending both level and range commands as a set, within the same SCPI message. For example,

```
VOLTage 10;VOLTage:RANGe 10<NL>
```

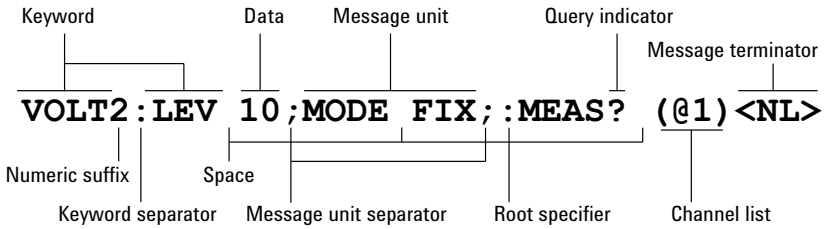
will always be correct because the commands are not executed until the message terminator is received. Because the range and setting information is received as a set, no range/setting conflict occurs.

SCPI Messages

There are two types of SCPI messages, program and response.

- A program message consists of one or more properly formatted SCPI commands sent from the controller to the instrument. The message, which may be sent at any time, requests the instrument to perform some action.
- A response message consists of data in a specific SCPI format sent from the instrument to the controller. The instrument sends the message only when commanded by a program message "query."

The following figure illustrates the SCPI message structure.



Headers

Headers, also referred to as keywords, are instructions recognized by the instrument. Headers may take a long form or a short form. In the long form, the header is completely spelled out, such as VOLTAGE, STATUS, and DELAY. In the short form, the header includes only the first three or four letters, such as VOLT, STAT, and DEL.

When the long form notation is used in this document, the capital letters indicate the corresponding short form. For example, when MEASure is the long form, MEAS will be the short form equivalent.

Message Unit

The simplest form of an SCPI command is a single message unit consisting of a command header (or keyword) followed by a message terminator such as a newline. The message unit may include a parameter after the header. The parameter can be a numeric value or a string.

```
*RST<NL>
VOLTage 20<NL>
```

Message Unit Separator

When two or more message units are combined into a compound message, separate the units with semicolons.

```
STATus:OPERation?;QUEStionable?
```

Root Specifier

When it precedes the first header of a message unit, a colon is interpreted as a root specifier. It tells the command parser that this is the root or the top node of the command tree.

Query Indicator

Following a header with a question mark turns it into a query (VOLTage?, VOLTage:TRIGgered?). The ? is the query indicator. If a query contains parameters, place the query indicator at the end of the last header, before the parameters.

```
VOLTage:TRIGgered? MAX
```

Message Terminator

A terminator informs SCPI that it has reached the end of a message. The following message terminators are permitted.

- newline <NL>, expressed in ASCII as decimal 10 or hex 0A
- end or identify <END> (EOI with ATN false)
- both of the above <NL><END>
- also <CR><NL>

In the examples used in this document, there is an assumed message terminator at the end of each message.

Numeric Suffix

All command headers can be accompanied by a numeric suffix for differentiating multiple instances of the same structure, i.e. for multi-channel instruments. The numeric suffix can be appended to both long and short forms. For example, TRIG1 is the short form of TRIGger1. A numeric suffix of 1 is implied on all command headers that do not explicitly define a suffix; thus, TRIG is equivalent to TRIG1.

For B2900 SCPI commands, some commands have a numeric suffix for classifying the channels, the limit tests, the Digital I/O pins, etc. The numeric suffix is an optional character, and can be expressed by [*c*], [*d*], [*h*], [*m*], and [*n*], where:

c is the integer 1 or 2, used to specify the channel 1 or 2, respectively.

d is the integer 1 or 2, used to specify the upper half or lower half, respectively, of the display area on the front panel display.

h is an integer between 1 and 100, used to specify a variable used in the program memory.

m is an integer between 1 and 12, used to specify a limit test.

n is an integer between 1 and 14, used to specify a Digital I/O pin.

For example, the :CALCulate[*c*]:LIMit[*m*]:STATe command allows you to enter *c* between :CALCulate and :LIMit, and *m* between :LIMit and :STATe.

Abbreviating the numeric suffix gives the same result as specifying 1.

Channel List Parameter

The channel list parameter is used for identifying the channel number as well as the numeric suffix.

The notation `(@1,2)` specifies a channel list that includes channels 1 and 2.

The notation `(@1:2)` specifies a channel list that includes channels 1 to 2.

In the B2900 SCPI commands, the channel list parameter is only available on certain commands which requires synchronization of channels (e.g. some commands of the TRIGger Subsystem) or specification of the channel itself (e.g. some commands of the MMEMory Subsystem).

The channel list parameter is also used for identifying the grouped channels defined by the :SYSTem:GROup[:DEFine] command for performing synchronous channel operations.

SCPI Command Completion

SCPI commands sent to the instrument are processed either sequentially or in parallel. Sequential commands finish execution before the subsequent command is started. Parallel commands allow other commands to begin executing while the parallel command is still executing.

The *WAI, *OPC, and *OPC? common commands provide different ways of indicating when all transmitted commands, including any parallel ones, have completed their operations. Some practical considerations for using these commands are as follows:

*WAI - prevents the instrument from processing subsequent commands until all pending operations are completed.

*OPC? - places a 1 in the Output Queue when all pending operations have completed. Since it requires your program to read the returned value before executing the next program statement, *OPC? can be used to cause the controller to wait for commands to complete before proceeding with its program.

*OPC - sets the OPC status bit when all pending operations have completed. Since your program can read this status bit on an interrupt basis, *OPC allows subsequent commands to be executed.

NOTE: The trigger subsystem must be in the Idle state for the status OPC bit to be true. As far as triggers are concerned, OPC is false whenever the trigger subsystem is in the Initiated state.

Device Clear

You can send a Device Clear at any time to abort an SCPI command that may be hanging up the GPIB interface. Device Clear aborts all transient and acquire actions, clears the input and output buffers of the instrument and prepares the instrument to accept a new command string. The error queue and all configuration states are left unchanged by Device Clear.

SCPI Conventions and Data Formats

The SCPI conventions shown in Table 1-1 are used throughout this document.

Data programmed or queried from the instrument is coded in ASCII. The data may contain numeric values or character strings.

Table 1-1 SCPI Conventions and Data Formats

Convention	Description
Angle brackets < >	Items within angle brackets are parameter abbreviations. For example, <NR1> indicates a specific form of numerical data.
Vertical bar |	Vertical bars separate alternative parameters. For example, VOLT | CURR indicates that either “VOLT” or “CURR” can be used as a parameter.
Square brackets [ ]	Items within square brackets are optional. The representation [SOURce:]VOLTage means that SOURce: may be omitted.
Parentheses ( )	Items within parentheses are used in place of the usual parameter types to specify a channel list. The notation (@1:3) specifies a channel list that includes channels 1, 2, and 3. The notation (@1,3) specifies a channel list that includes only channels 1 and 3.
Braces { }	Braces indicate parameters that may be repeated zero or more times. It is used especially for representing arrays. The notation <A>{,<B>} shows that parameter “A” must be entered, while parameter “B” omitted or may be entered one or more times.
<NR1>	Digits with an implied decimal point assumed at the right of the least-significant digit. Example: 273
<NR2>	Digits with an explicit decimal point. Example: 27.3
<NR3>	Digits with an explicit decimal point and an exponent. Example: 2.73E+02
<NRf>	Extended format that includes <NR1>, <NR2> and <NR3>. Examples: 273, 27.3, 2.73E+02

Convention	Description
<NRf+>	Expanded decimal format that includes <NRf>, MIN, and MAX. Examples: 273, 27.3, 2.73E+02, MAX. MIN and MAX are the minimum and maximum limit values that are implicit in the range specification for the parameter.
<NDN>	Non-decimal numeric value. May also be represented in binary preceded by “#B”, octal preceded by “#Q”, or hexadecimal preceded by “#H”. Examples: 29 (decimal), #B11101 (binary), #Q35 (octal), #H1D (hexadecimal)
<Bool>	Boolean data. Can be numeric (0, 1), or named (OFF, ON).
<SPD>	String program data. Programs string parameters enclosed in single or double quotes.
<CPD>	Character program data. Programs discrete parameters. Accepts both short form and long form.
<SRD>	String response data. Returns string parameters enclosed in single or double quotes.
<CRD>	Character response data. Returns discrete parameters. Only the short form of the parameter is returned.
<AARD>	Arbitrary ASCII response data. Permits the return of un-delimited 7-bit ASCII. This data type has an implied message terminator.
<Block>	Arbitrary block response data. Permits the return of definite length and indefinite length arbitrary response data. This data type has an implied message terminator.
<Expr>	Channel list, group list, or math expression. Channel list: Parenthetical data beginning with “@” Group list: Parenthetical data beginning with “@” Math expression: Parenthetical math expression (see :CALCulate:MATH[:EXPRession][:DEFine] command)

Data Output Format

B2900 supports the following data output formats for sending the result data. The data contains all of the elements specified by the :FORMat:ELEMents:SENSe or :FORMat:ELEMents:CALCulate command. Available elements are voltage measurement data, current measurement data, resistance measurement data, calculation result data, time data, status data, and source output setting data. A terminator <newline> (0x0a, 1 byte) is attached to the end of each data.

- ASCII data format, set by :FORMat[:DATA] ASCii

  Returns the result data in the comma-separated format. If the data contains three elements, B2900 sends the data as shown in the following example.

  Example: +1.000001E-06,+1.000002E-06,+9.999999E-07<newline>

  +9.910000E+37 indicates “not a number”.

  +9.900000E+37 indicates positive infinity.

  -9.900000E+37 indicates negative infinity.

- IEEE-754 single precision format, set by :FORMat[:DATA] REAL,32

  4-byte definite length block data, #<number of digits for byte length><byte length><byte>...<byte><terminator>. For example, two data elements are sent by a data block which consists of a header (3 bytes, #18), two 4-byte data, and a terminator (1 byte). A 4-byte data is used for each data element. Each element consists of a fraction (bits 0 (LSB) to 22), exponent (bits 23 to 30), and sign (bit 31).

  Order of bytes set by :FORMat:BORDer NORMal (default): byte 1 to 4

  Order of bytes set by :FORMat:BORDer SWAPped: byte 4 to 1

  NaN indicates “not a number”.

  +infinity indicates positive infinity.

  -infinity indicates negative infinity.

- IEEE-754 double precision format, set by :FORMat[:DATA] REAL,64

  8-byte definite length block data, #<number of digits for byte length><byte length><byte>...<byte><terminator>. For example, one data element is sent by a data block which consists of a header (3 bytes, #18), one 8-byte data, and a

terminator (1 byte). An 8-byte data is used for each data element. Each element consists of a fraction (bits 0 (LSB) to 51), exponent (bits 52 to 62), and sign (bit 63).

Order of bytes set by :FORMat:BORDer NORMal (default): byte 1 to 8

Order of bytes set by :FORMat:BORDer SWAPped: byte 8 to 1

NaN indicates “not a number”.

+infinity indicates positive infinity.

-infinity indicates negative infinity.

NOTE

If the conventional command set is used

If the IEEE-754 single precision data output format is used with the conventional command set, the output data will be an indefinite length block data, #0<byte length><byte>...<byte><terminator>. For example, two data elements are sent by a data block which consists of a header (2 bytes, #0), two 4-byte data, and a terminator (1 byte). A block is used for each data element. Each element consists of a fraction (bits 0 (LSB) to 22), exponent (bits 23 to 30), and sign (bit 31).

For the conventional command set, see Chapter 6, “Using Your Existing Programs.”

Status Data

B2900 sends the status data with the result data if it is specified by the :FORMat:ELEMents:SENSe or :FORMat:ELEMents:CALCulate command.

The status data is given by a binary-weighted sum of all bits set in the binary data. For example, if bit 3 (decimal value = 8) and bit 5 (decimal value = 32) are set to 1, the status data returns 40.

Bit definitions of the status data are shown in Table 1-2.

Table 1-2 Bit Definitions of Status Data

Bit	Description	Decimal value
0	0: Voltage source 1: Current source	0 or 1
1 and 2	Compliance condition 0: No or 1, 2, 3: Yes	0, 2, 4, or 6
3	Over voltage condition 0: No or 1: Yes	0 or 8
4	Over current condition 0: No or 1: Yes	0 or 16
5	High temperature condition 0: No or 1: Yes	0 or 32
13	Measurement range overflow 0: No or 1: Yes	0 or 8192
14	Offset compensation enable condition 0: No or 1: Yes	0 or 16384
15	Not used	0 or 32768
16 to 20	Composite limit test result, 0 to 31 See Table 1-3 for the Sorting mode test result and Table 1-4 for the Grading mode test result.	0 or (1 to 31)×2^{16} (0 or 65536 to 2031616)

Table 1-3 Composite Limit Test Result Bit Definitions for Sorting Mode

Bit20	Bit19	Bit18	Bit17	Bit16	Result
0	0	0	0	1	Limit 1: Passed limit test or failed compliance test
0	0	0	1	0	Limit 2: Passed limit test or failed compliance test
0	0	0	1	1	Limit 3: Passed limit test or failed compliance test
0	0	1	0	0	Limit 4: Passed limit test or failed compliance test
0	0	1	0	1	Limit 5: Passed limit test or failed compliance test
0	0	1	1	0	Limit 6: Passed limit test or failed compliance test
0	0	1	1	1	Limit 7: Passed limit test or failed compliance test
0	1	0	0	0	Limit 8: Passed limit test or failed compliance test
0	1	0	0	1	Limit 9: Passed limit test or failed compliance test
0	1	0	1	0	Limit 10: Passed limit test or failed compliance test
0	1	0	1	1	Limit 11: Passed limit test or failed compliance test
0	1	1	0	0	Limit 12: Passed limit test or failed compliance test
1	1	1	1	1	Passed compliance test and failed all limit tests

Table 1-4 Composite Limit Test Result Bit Definitions for Grading Mode

Bit20	Bit19	Bit18	Bit17	Bit16	Result
0	0	0	0	0	Passed all limit tests
1	0	0	0	1	Limit 1: Failed upper limit
0	0	0	0	1	Limit 1: Failed lower limit or failed compliance test
1	0	0	1	0	Limit 2: Failed upper limit
0	0	0	1	0	Limit 2: Failed lower limit or failed compliance test
1	0	0	1	1	Limit 3: Failed upper limit
0	0	0	1	1	Limit 3: Failed lower limit or failed compliance test
1	0	1	0	0	Limit 4: Failed upper limit
0	0	1	0	0	Limit 4: Failed lower limit or failed compliance test
1	0	1	0	1	Limit 5: Failed upper limit
0	0	1	0	1	Limit 5: Failed lower limit or failed compliance test
1	0	1	1	0	Limit 6: Failed upper limit
0	0	1	1	0	Limit 6: Failed lower limit or failed compliance test
1	0	1	1	1	Limit 7: Failed upper limit
0	0	1	1	1	Limit 7: Failed lower limit or failed compliance test
1	1	0	0	0	Limit 8: Failed upper limit
0	1	0	0	0	Limit 8: Failed lower limit or failed compliance test
1	1	0	0	1	Limit 9: Failed upper limit
0	1	0	0	1	Limit 9: Failed lower limit or failed compliance test
1	1	0	1	0	Limit 10: Failed upper limit
0	1	0	1	0	Limit 10: Failed lower limit or failed compliance test
1	1	0	1	1	Limit 11: Failed upper limit
0	1	0	1	1	Limit 11: Failed lower limit or failed compliance test
1	1	1	0	0	Limit 12: Failed upper limit
0	1	1	0	0	Limit 12: Failed lower limit or failed compliance test

GPIB Capability

The following table lists the GPIB capabilities and functions of the B2900. These functions provide the means for an instrument to receive, process, and transmit, commands, data, and status over the GPIB bus.

Interface Function	Code	Description
Source Handshake	SH1	Complete capability
Acceptor Handshake	AH1	Complete capability
Talker	T6	Basic Talker: YES Serial Poll: YES Talk Only Mode: NO Unaddress if MLA (my listen address): YES
Listener	L4	Basic Listener: YES Unaddress if MTA (my talk address): YES Listen Only Mode: NO
Service Request	SR1	Complete capability
Remote/Local	RL1	Complete capability (with local lockout)
Parallel Poll	PP0	No capability
Device Clear	DC1	Complete capability
Device Trigger	DT1	Complete capability
Controller Function	C0	No capability
Driver Electronics	E1	Open Collector

Status Byte

Status byte bits are turned off or on (0 or 1) to represent the instrument operation status. When you execute a serial poll, an external computer (controller) reads the contents of the status byte, and responds accordingly. When an unmasked status bit is set to “1”, the instrument sends an SRQ to the controller, causing the controller to perform an interrupt service routine.

Bit	Decimal Value	Description
0	1	Measurement status summary
1	2	Not used
2	4	Error queue not empty
3	8	Questionable status summary
4	16	Output buffer
5	32	Event status byte summary
6	64	Master status summary (Request for service)
7	128	Operation status summary

The status byte register can be read with either a serial poll or the *STB? query command. Serial poll is a low-level GPIB command.

In general, use serial polling (not *STB?) inside interrupt service routines. Use *STB? in other cases (not in interrupt service routine) when you want to know the value of the Status Byte.

Status System Diagram

- Figure 1-1, “B2900 Status System Overview.”
- Figure 1-2, “Measurement Status register.”
- Figure 1-3, “Questionable Status register.”
- Figure 1-4, “Standard Event Status register.”
- Figure 1-5, “Operation Status register.”

Figure 1-1 B2900 Status System Overview

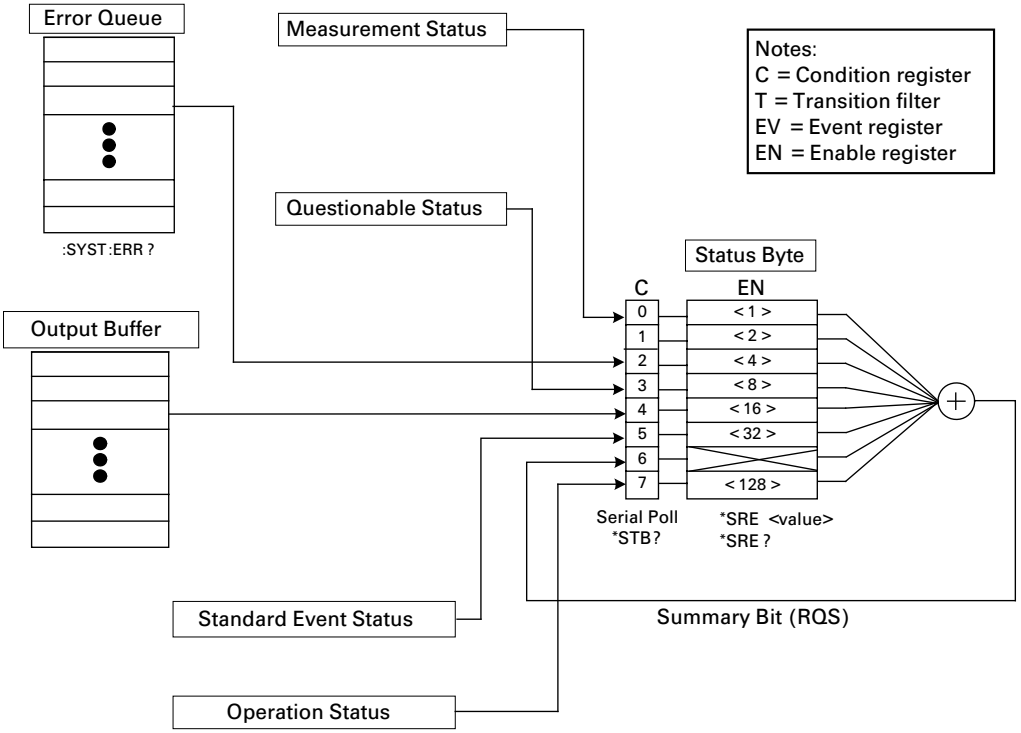


Figure 1-2 **Measurement Status register**

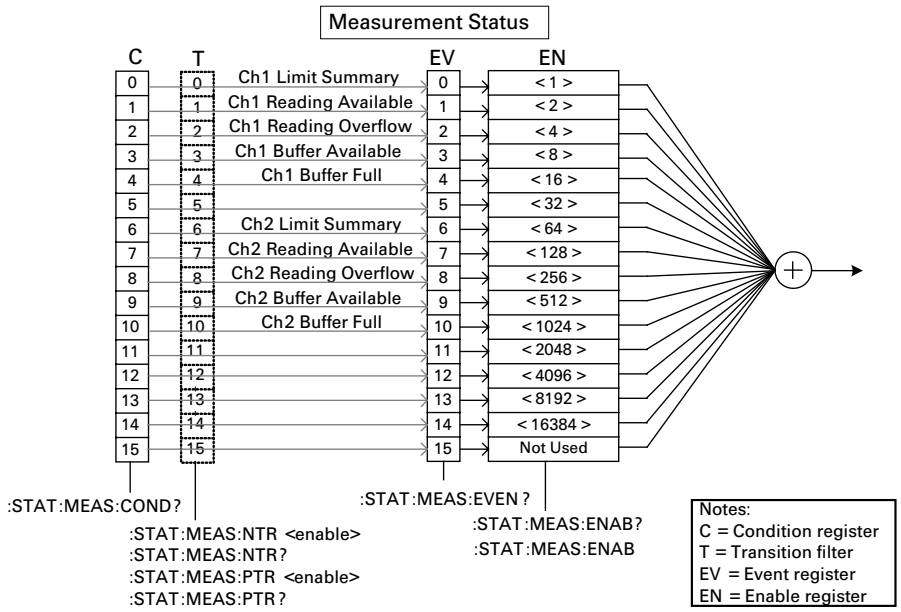


Figure 1-3 Questionable Status register

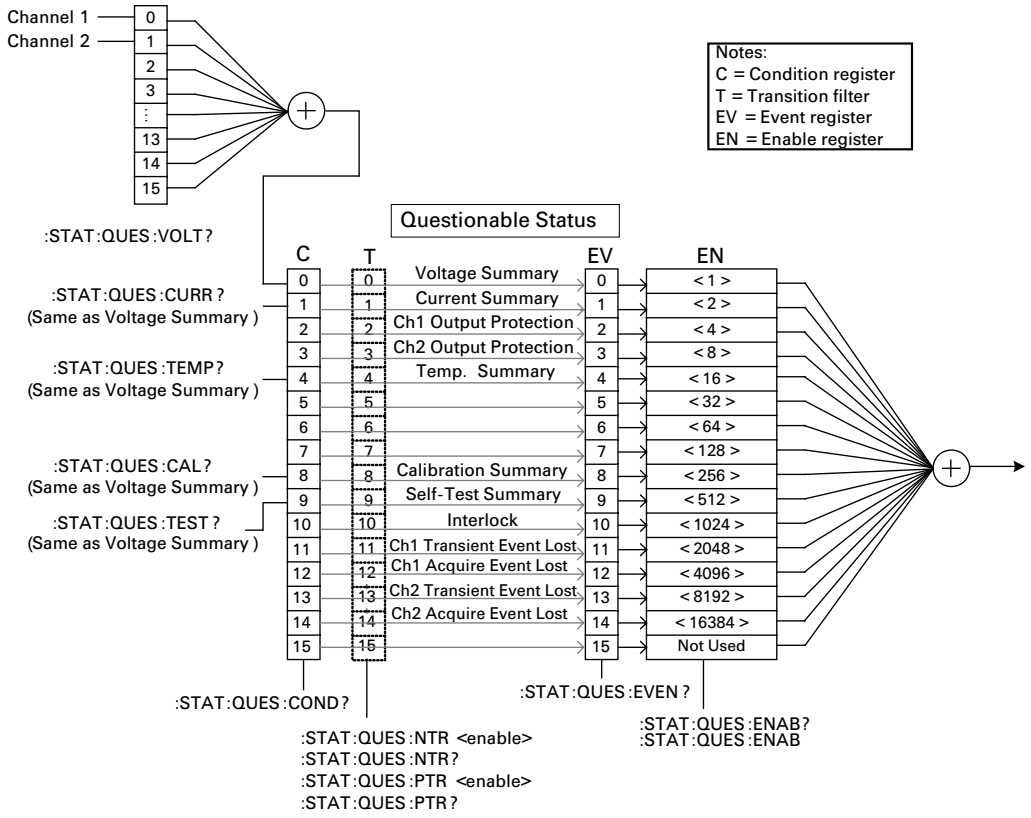


Figure 1-4 **Standard Event Status register**

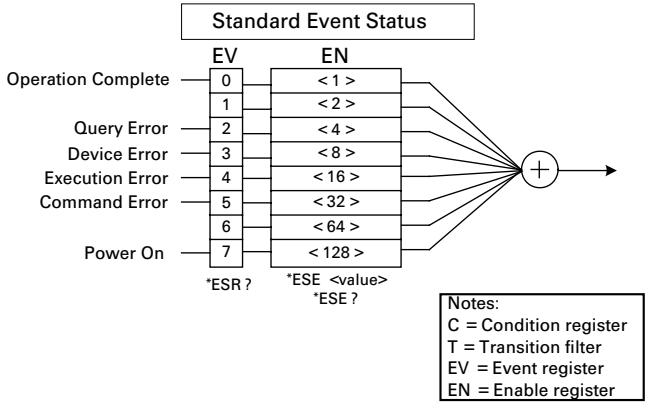
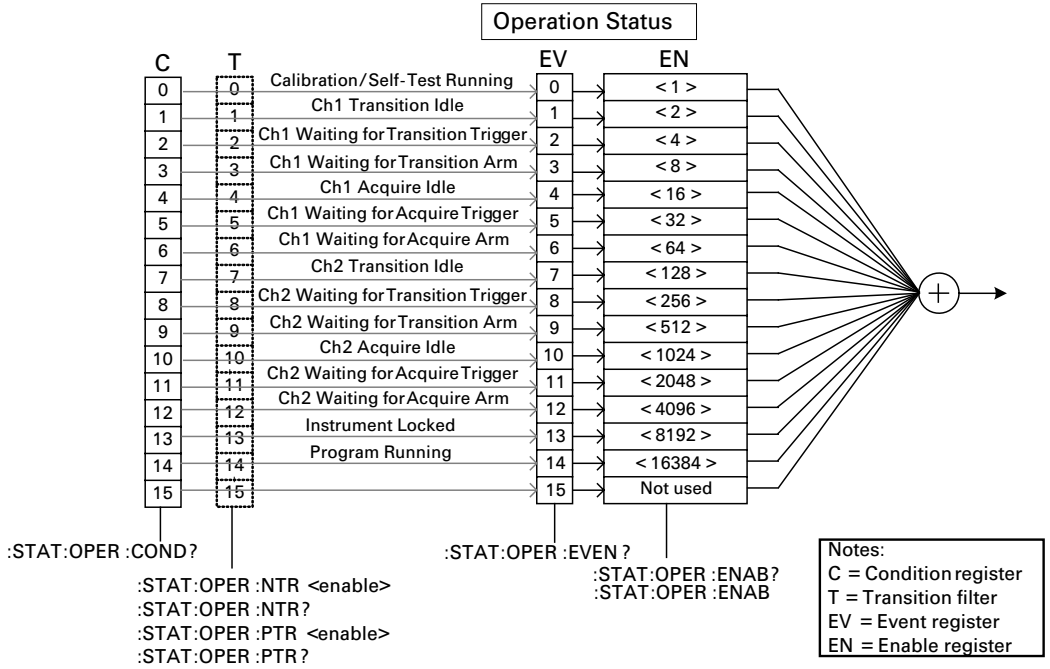


Figure 1-5 **Operation Status register**



Non-Volatile Settings

The following tables show the factory-shipped non-volatile settings of the instrument. Information in non-volatile memory is NOT lost when power is turned off. These settings are all customer-configurable.

Table 1-5 Non-volatile Communication Settings

Setup item	Factory default setting
DHCP	Enabled
IP address	169.254.5.2
Subnet mask	255.255.0.0
Default gateway	0.0.0.0
Obtain DNS server from DHCP	Enabled
DNS server	0.0.0.0
WINS server	0.0.0.0
Hostname	A-B29*xx*A-*nnnnn* B29*xx*A: model number *nnnnn*: suffix of serial number.
Desired hostname	
Desired service name	
mDNS	Enabled
Use DNS naming service	Enabled
Use NetBIOS naming service	Enabled
Domain name	Not set
GPIB address	23
LXI identify	Disabled
GPIB command interface	Enabled
USB command interface	Enabled
VXI-11 command interface	Enabled

Setup item	Factory default setting
SCPI telnet command interface	Enabled
SCPI socket command interface	Enabled
SCPI HiSLIP command interface	Enabled
Web interface	Enabled
Command prompt for a Telnet session	B2900A>
Welcome message for a Telnet session	Welcome to Agilent B2900A Series

Table 1-6 **Other Non-volatile Settings**

Setup item	Factory default setting
Channel grouping	“1” for 1-ch models “1-2” for 2-ch models
Remote display	Enabled
Display color set	1
Beeper	Enabled
Web server	Enabled
SCPI language mode	Default
Power-on program	Not set
Line frequency	50 Hz
Fan control mode	Normal

2 Subsystem Command Summary

This chapter lists all of the SCPI subsystem commands for Agilent B2900 and provides the summary information of the command.

- "Setting Source/Measure Unit"

  "SOURce Subsystem," for source setup

  "SENSe Subsystem," for measurement setup

  "OUTPut Subsystem," for using source output functions

  "Source Output Ranges"

  "Measurement Ranges"
- "Controlling Source/Measure Trigger"

  "TRIGger Subsystem," for triggering source output and measurement
- "Reading Source/Measure Data"

  "FETCh Subsystem," only for reading data

  "FORMat Subsystem," for data output format

  "READ Subsystem," for performing measurements

  "MEASure Subsystem," for a spot measurement

  "CALCulate Subsystem," for using limit test, calculation, and math functions

  "TRACe Subsystem," for using trace buffer
- "Using Advanced Functions"

  "HCOPy Subsystem," for getting screen dump

  "DISPlay Subsystem," for front panel display setup

  "MMEMory Subsystem," for managing data memory

  "PROGram Subsystem," for using program memory

  "SYSTem Subsystem," for using system functions

  "STATus Subsystem," for using status system

NOTE In the tables, Reset setting gives the initial setting or the default setting which is set to the instrument when it is turned on or it receives the *RST command.

NOTE

All commands described in this chapter are effective for the default remote control mode which supports all B2900 functions. B2900 uses this mode if it is not changed since the instrument is shipped from the factory.

You may want to use existing programs for controlling existing instruments, such as Series 2400 from Keithley Instruments, Inc. To do so, refer to Chapter 6, “Using Your Existing Programs.” B2900 can support many of the conventional SCPI commands by switching the remote control mode.

NOTE

The following subsystem commands are classified under the TRIGger subsystem because they are used for trigger operations.

- ABORt
- ARM
- IDLE
- INITiate
- TRIGger

For examples of the SMU source output and measurement operation, see Figure 2-7. Also see Figures 2-6 and 2-8 for using the trigger commands. For these commands, see Table 2-10.

NOTE

For details on numeric suffixes [*c*], [*d*], [*h*], [*m*], and [*n*], see “Numeric Suffix” on page 1-8.

Setting Source/Measure Unit

Figure 2-1 **To Perform Staircase Sweep Measurement**

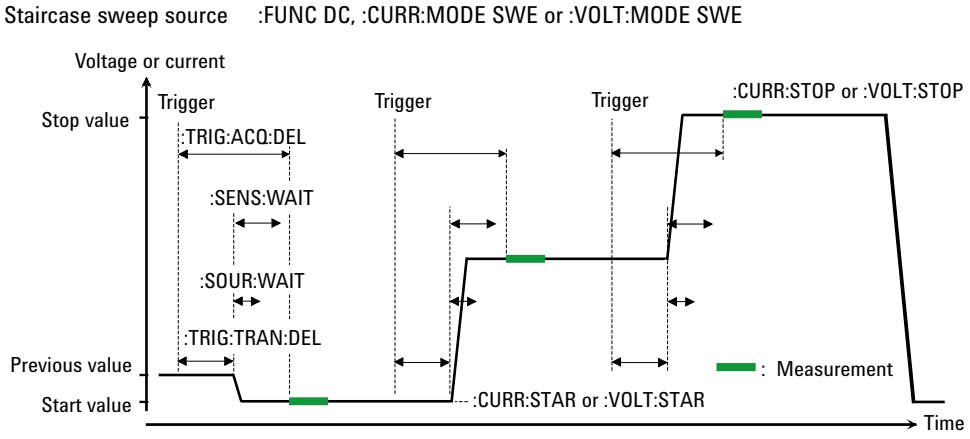


Figure 2-2 **To Perform Pulsed Sweep Measurement**

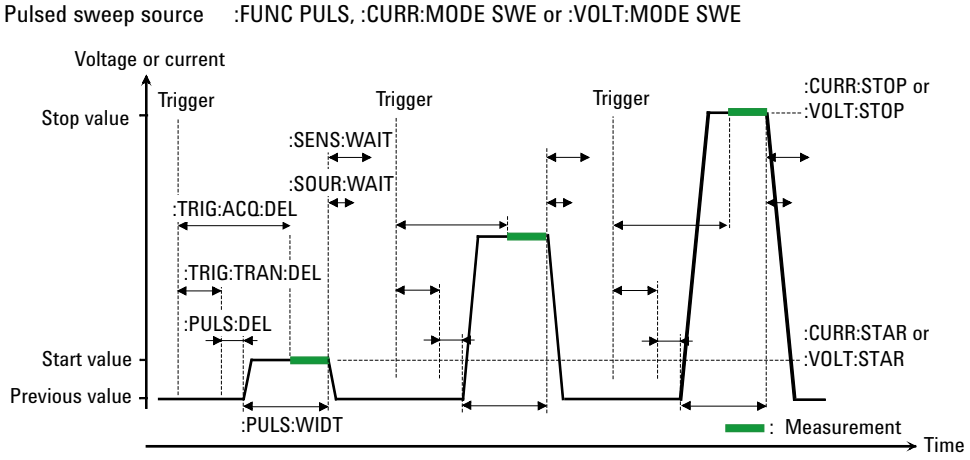


Figure 2-3 **Variety of Sweep Outputs**

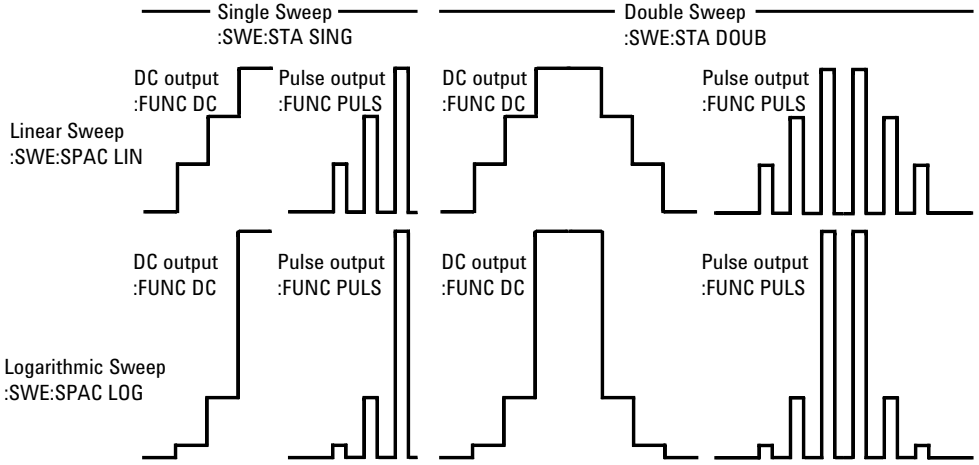


Figure 2-4 **To Perform Spot Measurement**

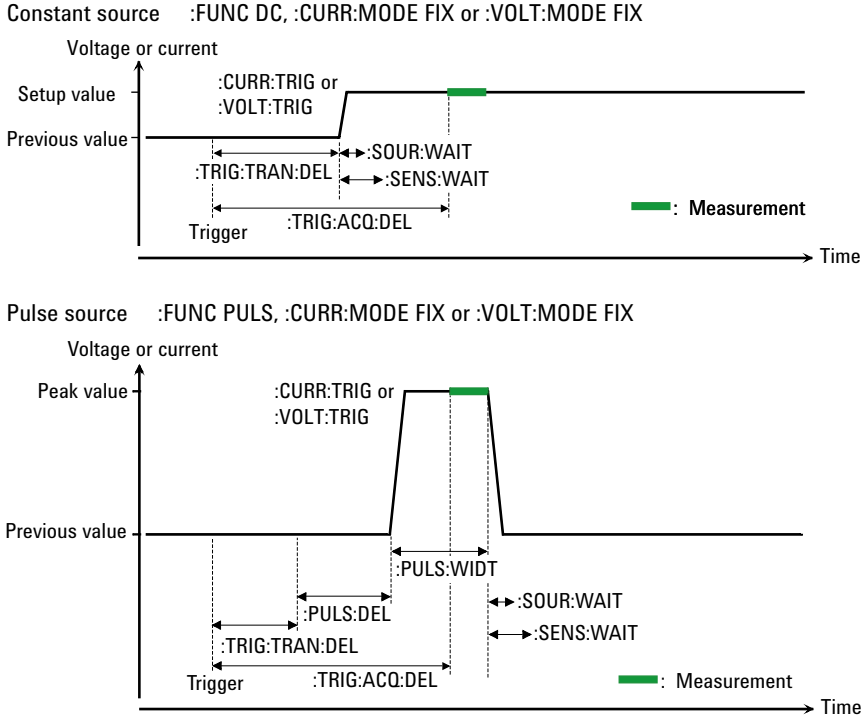


Table 2-1 SOURce Subsystem

Command	Summary	Reset setting

`[:SOUR[c]]:CURR:CENT` *data*
`[:SOUR[c]]:VOLT:CENT` *data*
`[:SOUR[c]]:CURR:CENT? [DEFault|MINimum|MAXimum]`
`[:SOUR[c]]:VOLT:CENT? [DEFault|MINimum|MAXimum]`
	Sets the center or span value of the current or voltage sweep output.
	data=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.
	The center and span values can be expressed by the following formula, using the start and stop values set by the [:SOUR[*c*]]:<CURR|VOLT>:<STAR|STOP> command. So the last command setting is effective for these sweep parameters.
	center = (*start* + *stop*)/2
	span = *stop* - *start*
		0

`[:SOUR[c]]:CURR:SPAN` *data*
`[:SOUR[c]]:VOLT:SPAN` *data*
`[:SOUR[c]]:CURR:SPAN? [DEFault|MINimum|MAXimum]`
`[:SOUR[c]]:VOLT:SPAN? [DEFault|MINimum|MAXimum]`
		0

`[:SOUR[c]]:CURR[:LEV][:IMM][:AMPL]` *level*
`[:SOUR[c]]:VOLT[:LEV][:IMM][:AMPL]` *level*
`[:SOUR[c]]:CURR[:LEV][:IMM][:AMPL]? [DEFault|MINimum|MAXimum]`
`[:SOUR[c]]:VOLT[:LEV][:IMM][:AMPL]? [DEFault|MINimum|MAXimum]`
	Changes the output level of the specified source channel immediately.
	level=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.
		0

`[:SOUR[c]]:CURR[:LEV]:TRIG[:AMPL]` *level*
`[:SOUR[c]]:VOLT[:LEV]:TRIG[:AMPL]` *level*
`[:SOUR[c]]:CURR[:LEV]:TRIG[:AMPL]? [DEFault|MINimum|MAXimum]`
`[:SOUR[c]]:VOLT[:LEV]:TRIG[:AMPL]? [DEFault|MINimum|MAXimum]`
	Changes the output level of the specified source channel immediately by receiving the trigger source set by the :TRIG[c]<:ACQ|:TRAN[:ALL]>:SOUR command.
	level=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.
		0

Command	Summary	Reset setting
[:SOUR[*c*]]:CURR:MODE *mode* [:SOUR[*c*]]:VOLT:MODE *mode* [:SOUR[*c*]]:CURR:MODE? [:SOUR[*c*]]:VOLT:MODE?	Selects the source mode, fixed, list sweep, or sweep, of the specified source channel. *mode*=SWEep|LIST|FIXed	FIX
[:SOUR[*c*]]:CURR:POIN *points* [:SOUR[*c*]]:VOLT:POIN *points* [:SOUR[*c*]]:CURR:POIN? [DEFault|MINimum|MAXimum] [:SOUR[*c*]]:VOLT:POIN? [DEFault|MINimum|MAXimum]	Sets the number of sweep steps for the current or voltage sweep output. *points*=MINimum|MAXimum|DEFault|1 to 2500 The points value can be expressed by the following formula, using the step value set by the [:SOUR[*c*]]:<CURR|VOLT>:STEP command and the span value set by the [:SOUR[*c*]]:<CURR|VOLT>:SPAN command. *points* = *span*/*step* + 1 (where *step* is not 0) *points*=1 sets *step*=0.	1
[:SOUR[*c*]]:CURR:RANG *range* [:SOUR[*c*]]:VOLT:RANG *range* [:SOUR[*c*]]:CURR:RANG? [:SOUR[*c*]]:VOLT:RANG?	Sets the current or voltage output range of the specified source channel. This command is effective when the automatic ranging function is off. *range*=MINimum|MAXimum|DEFault|minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.	1.00E-04 A for current range, or 2 V for voltage range
[:SOUR[*c*]]:CURR:RANG:AUTO *mode* [:SOUR[*c*]]:VOLT:RANG:AUTO *mode* [:SOUR[*c*]]:CURR:RANG:AUTO? [:SOUR[*c*]]:VOLT:RANG:AUTO?	Enables or disables the automatic ranging function for the specified source channel. *mode*=1|ON|0|OFF	ON

Command	Summary	Reset setting
[:SOUR[*c*]]:CURR:RANG:AUTO:LLIM *range* [:SOUR[*c*]]:VOLT:RANG:AUTO:LLIM *range* [:SOUR[*c*]]:CURR:RANG:AUTO:LLIM? [DEFault|MINimum|MAXimum] [:SOUR[*c*]]:VOLT:RANG:AUTO:LLIM? [DEFault|MINimum|MAXimum]	Specifies the lower limit for the automatic output ranging operation, and sets the minimum range which provides the best resolution to apply the specified value. *range*=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See "Source Output Ranges" on page 2-24.	1.00E-06 A for current range, or 0.2 V for voltage range
[:SOUR[*c*]]:CURR:STAR *data* [:SOUR[*c*]]:VOLT:STAR *data* [:SOUR[*c*]]:CURR:STAR? [DEFault|MINimum|MAXimum] [:SOUR[*c*]]:VOLT:STAR? [DEFault|MINimum|MAXimum]	Sets the start or stop value for the current or voltage sweep output. *data*=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See "Source Output Ranges" on page 2-24. The start and stop values can be expressed by the following formula, using the center and span values set by the [:SOUR[*c*]]:<CURR|VOLT>:<CENT|SPAN> command. So the last command setting is effective for these sweep parameters. *start* = *center* - *span*/2 *stop* = *center* + *span*/2	0
[:SOUR[*c*]]:CURR:STOP *data* [:SOUR[*c*]]:VOLT:STOP *data* [:SOUR[*c*]]:CURR:STOP? [DEFault|MINimum|MAXimum] [:SOUR[*c*]]:VOLT:STOP? [DEFault|MINimum|MAXimum]		0

Command	Summary	Reset setting
[:SOUR[*c*]]:CURR:STEP *step* [:SOUR[*c*]]:VOLT:STEP *step* [:SOUR[*c*]]:CURR:STEP? [DEFault|MINimum|MAXimum] [:SOUR[*c*]]:VOLT:STEP? [DEFault|MINimum|MAXimum]	Sets the sweep step value of the current or voltage sweep output. *step*=MINimum|MAXimum|DEFault| minimum to maximum source value, in A or V. See "Source Output Ranges" on page 2-24. The step value can be expressed by the following formula, using the points value set by the [:SOUR[*c*]]:<CURR|VOLT>:POIN command and the span value set by the [:SOUR[*c*]]:<CURR|VOLT>:SPAN command. *step* = *span*/(*points* - 1) (where *points* is not 1) *points*=1 sets *step*=0.	0
[:SOUR]:DIG:DATA *data* [:SOUR]:DIG:DATA?	Sets the output data to the GPIO pins (digital control port) and read data from the GPIO pins. *data*=0 to 16383	
[:SOUR]:DIG:EXT[*n*][:FUNC] *function* [:SOUR]:DIG:EXT[*n*][:FUNC]?	Assigns the input/output function to the specified GPIO pin. *function*=DIO|DINPut|HVOL|TINPut| TOUT	DINP for EXT1 to 13, HVOL for EXT14
[:SOUR]:DIG:EXT[*n*]:POL *polarity* [:SOUR]:DIG:EXT[*n*]:POL?	Sets the polarity of the input/output function for the specified GPIO pin. *polarity*=NEG|POS	NEG for EXT1 to 13, POS for EXT14
[:SOUR]:DIG:EXT[*n*]:TOUT[:EDGE]:POS *position* [:SOUR]:DIG:EXT[*n*]:TOUT[:EDGE]:POS?	Selects the output trigger timing for the specified GPIO pin. *position*=BEFore|AFTer|BOTH	BOTH

Command	Summary	Reset setting
`[:SOUR]:DIG:EXT[n]:TOUT[:EDGE]:WIDT width` `[:SOUR]:DIG:EXT[n]:TOUT[:EDGE]:WIDT? [DEFault|MINimum|MAXimum]`	Sets the pulse width of the output trigger for the specified GPIO pin. *width*=MINimum|MAXimum|DEFault|1E-5 to 1E-2 seconds	1.00E-04 seconds
`[:SOUR]:DIG:EXT[n]:TOUT:TYPE type` `[:SOUR]:DIG:EXT[n]:TOUT:TYPE?`	Selects the output trigger type for the specified GPIO pin. *type*=EDGE|LEVel	EDGE
`[:SOUR]:DIG:INT[n]:TOUT[:EDGE]:POS position` `[:SOUR]:DIG:INT[n]:TOUT[:EDGE]:POS?`	Selects the output trigger timing for the specified channel. *position*=BEFore|AFTer|BOTH	BOTH
`[:SOUR[c]]:FUNC:MODE mode` `[:SOUR[c]]:FUNC:MODE?`	Selects the source output mode of the specified channel. *mode*=CURRent|VOLTage	VOLT
`[:SOUR[c]]:FUNC[:SHAP] shape` `[:SOUR[c]]:FUNC[:SHAP]?`	Selects the source output shape of the specified channel. *shape*=PULSe|DC	DC
`[:SOUR[c]]:FUNC:TRIG:CONT mode` `[:SOUR[c]]:FUNC:TRIG:CONT?`	Enables or disables continuous trigger output for the specified channel. *mode*=0|OFF|1|ON	OFF

`[:SOUR[c]]:LIST:CURR list`
`[:SOUR[c]]:LIST:VOLT list`
`[:SOUR[c]]:LIST:CURR?`
`[:SOUR[c]]:LIST:VOLT?`
	Sets the source output current or voltage data for the specified channel.
	list: List of the output current or voltage data. Maximum of 2500 data can be set to *list*. Each data must be separated by a comma.
		0

Command	Summary	Reset setting
`[:SOUR[c]]:LIST:CURR:APP append_list` `[:SOUR[c]]:LIST:VOLT:APP append_list`	Adds the source output current or voltage data to the end of the list set by the [:SOUR[*c*]]:LIST:<CURR|VOLT> command, to which some data might be appended to by this command. Total number of data in the list must be ≤ 2500. *append_list*: List of the output current or voltage data. Multiple data can be set to *append_list*. Each data must be separated by a comma.	
`[:SOUR[c]]:LIST:CURR:POIN?` `[:SOUR[c]]:LIST:VOLT:POIN?`	Returns the number of data in the list set by the [:SOUR[*c*]]:LIST:<CURR|VOLT> command, to which some data might be appended to by the [:SOUR[*c*]]:LIST:<CURR|VOLT>:APP command.	

`[:SOUR[c]]:LIST:CURR:STAR start`
`[:SOUR[c]]:LIST:VOLT:STAR start`
`[:SOUR[c]]:LIST:CURR:STAR?`
`[:SOUR[c]]:LIST:VOLT:STAR?`
	Specifies the list sweep start point by using the index of the list.
	start: Index of the list. 1 to 2500. *start*=1 indicates the first data in the list (top of the list). *start*=0 or the value greater than 2500 causes an error.
		1

`[:SOUR[c]]:PULS:DEL delay` `[:SOUR[c]]:PULS:DEL? [DEFault|MINimum|MAXimum]`	Sets the pulse delay time for the specified channel. The pulse delay time is the time from starting the pulse base output to starting the pulse level transition (or to starting the pulse peak output). See Figures 2-4 and 2-2. *delay*=DEFault|MINimum|MAXimum| 0.0 to 99999.9 seconds	0

Command	Summary	Reset setting
`[:SOUR[c]]:PULS:WIDT` *width* `[:SOUR[c]]:PULS:WIDT? [DEFault|MINimum|MAXimum]`	Sets the pulse width for the specified channel. The pulse width is the time from starting the pulse peak output (or starting the pulse level transition) to the end of the pulse peak output.See Figures 2-2 and 2-4. *width*=DEFault|MINimum|MAXimum|5E-5 to 100000 seconds, in 1E-6 resolution Minimum time for the pulse base output is also 50 µs. And the minimum pulse period is 100 µs.	5E-5 seconds
`[:SOUR[c]]:SWE:DIR` *direction* `[:SOUR[c]]:SWE:DIR?`	Sets the sweep direction, UP or DOWN, for the specified channel. *direction*=DOWN|UP	UP
`[:SOUR[c]]:SWE:POIN` *points* `[:SOUR[c]]:SWE:POIN? DEFault|MINimum|MAXimum`	Sets the number of sweep steps for the specified channel. This command setting is effective for both current sweep and voltage sweep. *points*=DEFault|MINimum|MAXimum|1 to 2500 The points value can be expressed by the following formula, using the step value set by the [:SOUR[*c*]]:<CURR|VOLT>:STEP command and the span value set by the [:SOUR[*c*]]:<CURR|VOLT>:SPAN command. *points* = *span*/*step* + 1 (where *step* is not 0) *points*=1 sets *step*=0.	1

Command	Summary	Reset setting
[:SOUR[*c*]]:SWE:RANG *mode* [:SOUR[*c*]]:SWE:RANG?	Selects the output ranging mode of the sweep output for the specified channel. *mode*=BEST|FIXed|AUTO	BEST
[:SOUR[*c*]]:SWE:SPAC *mode* [:SOUR[*c*]]:SWE:SPAC?	Selects the scale of the sweep output for the specified channel. See Figure 2-3. *mode*=LOGarithmic|LINear	LIN
[:SOUR[*c*]]:SWE:STA *mode* [:SOUR[*c*]]:SWE:STA?	Sets the sweep mode for the specified channel. *mode*=SINGle|DOUBle	SING
[:SOUR[*c*]]:TOUT:SIGN *output*{,output} [:SOUR[*c*]]:TOUT:SIGN *output*{,output} [:SOUR[*c*]]:TOUT:SIGN *output*{,output} [:SOUR[*c*]]:TOUT:SIGN? [:SOUR[*c*]]:TOUT:SIGN?	Selects the trigger output for the status change between the trigger layer and the transient device action. *output*=INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	EXT1
[:SOUR[*c*]]:TOUT[:STAT] *mode* [:SOUR[*c*]]:TOUT[:STAT] *mode* [:SOUR[*c*]]:TOUT[:STAT] *mode* [:SOUR[*c*]]:TOUT[:STAT]? [:SOUR[*c*]]:TOUT[:STAT]?	Enables or disables the trigger output for the status change between the trigger layer and the transient device action. *mode*=1|ON|0|OFF	OFF
[:SOUR[*c*]]:WAIT:AUTO *mode* [:SOUR[*c*]]:WAIT:AUTO?	Enables or disables the initial wait time used for calculating the source wait time for the specified channel. See [:SOUR[*c*]]:WAIT[:STAT]. *mode*=1|ON|0|OFF	ON

Command	Summary	Reset setting
`[:SOUR[c]]:WAIT:GAIN gain` `[:SOUR[c]]:WAIT:GAIN? [DEFault|MINimum|MAXimum]`	Sets the gain value used for calculating the source wait time for the specified channel. *gain*=MINimum|MAXimum|DEFault| 0 to 100	1
`[:SOUR[c]]:WAIT:OFFS offset` `[:SOUR[c]]:WAIT:OFFS? [DEFault|MINimum|MAXimum]`	Sets the offset value used for calculating the source wait time for the specified channel. *gain*=MINimum|MAXimum|DEFault| 0 to 1 seconds	0
`[:SOUR[c]]:WAIT[:STAT] mode` `[:SOUR[c]]:WAIT[:STAT]?`	Enables or disables the source wait time for the specified channel. This wait time is defined as the time the source channel cannot change the output after the start of a DC output or the trailing edge of a pulse. *mode*=0 or OFF disables the source wait time. The wait time is set to 0. *mode*=1 or ON enables the source wait time given by the following formula. • [:SOUR[*c*]]:WAIT:AUTO ON|1 condition: wait time = *gain* × initial wait time + *offset* • [:SOUR[*c*]]:WAIT:AUTO OFF|0 condition: wait time = *offset* The initial wait time is automatically set by the instrument and cannot be changed.	ON

Table 2-2 SENSe Subsystem

Command	Summary	Reset setting
:SENS[*c*]:CURR[:DC]:APER *time* :SENS[*c*]:RES:APER *time* :SENS[*c*]:VOLT[:DC]:APER *time* :SENS[*c*]:CURR[:DC]:APER? [DEFault|MINimum|MAXimum] :SENS[*c*]:RES:APER? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:APER? [DEFault|MINimum|MAXimum]	Sets the integration time for one point measurement. *time*=MINimum|MAXimum|DEFault|+8E-6 to +2 seconds The integration time can be expressed by the following formula by using the NPLC value set by the :SENS:<CURR|RES|VOLT>:NPLC command. So the last command setting is effective for both *time* and *nplc*. *time = nplc / power line frequency*	0.1 PLC, =0.1/*power line frequency*
:SENS[*c*]:CURR[:DC]:APER:AUTO *mode* :SENS[*c*]:RES:APER:AUTO *mode* :SENS[*c*]:VOLT[:DC]:APER:AUTO *mode* :SENS[*c*]:CURR[:DC]:APER:AUTO? :SENS[*c*]:RES:APER:AUTO? :SENS[*c*]:VOLT[:DC]:APER:AUTO?	Enables or disables the automatic aperture function. *mode*=1|ON|0|OFF The automatic aperture on/off works with the automatic NPLC on/off set by the :SENS:<CURR|RES|VOLT>:NPLC:AUTO command. So the last command setting is effective for both functions.	ON
:SENS[*c*]:CURR[:DC]:NPLC *nplc* :SENS[*c*]:RES:NPLC *nplc* :SENS[*c*]:VOLT[:DC]:NPLC *nplc* :SENS[*c*]:CURR[:DC]:NPLC? [DEFault|MINimum|MAXimum] :SENS[*c*]:RES:NPLC? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:NPLC? [DEFault|MINimum|MAXimum]	Sets the number of power line cycles (NPLC) value instead of setting the integration time for one point measurement. *nplc*=MINimum|MAXimum|DEFault|+4E-4 to +100 for 50 Hz or +4.8E-4 to +120 for 60 Hz The NPLC value can be expressed by the following formula by using the integration time set by the :SENS:<CURR|RES|VOLT>:APER command. So the last command setting is effective for both *nplc* and *time*. *nplc = time × power line frequency*	0.1 PLC, =0.1/*power line frequency*

Command	Summary	Reset setting
:SENS[*c*]:CURR[:DC]:NPLC:AUTO *mode* :SENS[*c*]:RES:NPLC:AUTO *mode* :SENS[*c*]:VOLT[:DC]:NPLC:AUTO *mode* :SENS[*c*]:CURR[:DC]:NPLC:AUTO? :SENS[*c*]:RES:NPLC:AUTO? :SENS[*c*]:VOLT[:DC]:NPLC:AUTO?	Enables or disables the automatic NPLC function. *mode*=1|ON|0|OFF The automatic NPLC on/off works with the automatic aperture on/off set by the :SENS:<CURR|RES|VOLT>:APER:AUTO command. So the last command setting is effective for both functions.	OFF
:SENS[*c*]:CURR[:DC]:PROT[:LEV] *comp* :SENS[*c*]:VOLT[:DC]:PROT[:LEV] *comp* :SENS[*c*]:CURR[:DC]:PROT[:LEV]? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:PROT[:LEV]? [DEFault|MINimum|MAXimum]	Sets the compliance value of the specified channel. *comp*=MINimum|MAXimum|DEFault| minimum to maximum measurement value, in A or V. See “Measurement Ranges” on page 2-27.	1.00E-04 A for current compliance, 2.0 V for voltage compliance
:SENS[*c*]:CURR[:DC]:PROT:TRIP? :SENS[*c*]:VOLT[:DC]:PROT:TRIP?	Returns the compliance status of the specified channel Response is 1 or 0 that indicates the channel is in the compliance state or not.	
:SENS[*c*]:CURR[:DC]:RANG:AUTO *mode* :SENS[*c*]:RES:RANG:AUTO *mode* :SENS[*c*]:VOLT[:DC]:RANG:AUTO *mode* :SENS[*c*]:CURR[:DC]:RANG:AUTO? :SENS[*c*]:RES:RANG:AUTO? :SENS[*c*]:VOLT[:DC]:RANG:AUTO?	Enables or disables the automatic ranging function of the specified measurement channel. *mode*=1|ON|0|OFF	ON

Command	Summary	Reset setting
:SENS[*c*]:CURR[:DC]:RANG:AUTO:LLIM *range* :SENS[*c*]:RES:RANG:AUTO:LLIM *range* :SENS[*c*]:VOLT[:DC]:RANG:AUTO:LLIM *range* :SENS[*c*]:CURR[:DC]:RANG:AUTO:LLIM? [DEFault|MINimum|MAXimum] :SENS[*c*]:RES:RANG:AUTO:LLIM? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:RANG:AUTO:LLIM? [DEFault|MINimum|MAXimum]	Specifies the lower limit for the automatic measurement ranging operation, and sets the minimum measurement range which provides the best resolution to measure the specified value. *range*=MINimum|MAXimum|DEFault| minimum to maximum measurement value, in A, Ω, or V. See “Measurement Ranges” on page 2-27.	1.00E-06 A, 2 Ω, or 0.2 V
:SENS[*c*]:CURR[:DC]:RANG:AUTO:MODE *mode* :SENS[*c*]:VOLT[:DC]:RANG:AUTO:MODE *mode* :SENS[*c*]:CURR[:DC]:RANG:AUTO:MODE? :SENS[*c*]:VOLT[:DC]:RANG:AUTO:MODE?	Selects the operation mode of the automatic measurement ranging. This command setting is not effective if the automatic ranging is disabled by the :SENS:<CURR|VOLT>:RANG:AUTO command. *mode*=NORMal|RESolution|SPEed	NORM
:SENS[*c*]:CURR[:DC]:RANG:AUTO:THR *rate* :SENS[*c*]:VOLT[:DC]:RANG:AUTO:THR *rate* :SENS[*c*]:CURR[:DC]:RANG:AUTO:THR? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:RANG:AUTO:THR? [DEFault|MINimum|MAXimum]	Sets the threshold rate for the automatic measurement ranging operation. *rate*=MINimum|MAXimum|DEFault| 11 % to 100 %	90

Command	Summary	Reset setting
:SENS[*c*]:RES:RANG:AUTO:ULIM *range* :SENS[*c*]:RES:RANG:AUTO:ULIM? [DEFault|MINimum|MAXimum] :SENS[*c*]:CURR[:DC]:RANG:AUTO:ULIM? :SENS[*c*]:VOLT[:DC]:RANG:AUTO:ULIM?	Specifies the upper limit for the automatic measurement ranging operation, and sets the maximum measurement range which provides the best resolution to measure the specified value. This is effective for resistance measurements set to the AUTO mode by the :SENS:RES:MODE command. *range*=MINimum|MAXimum|DEFault| value (see Table 2-9)	200 MΩ
:SENS[*c*]:CURR[:DC]:RANG[:UPP] *range* :SENS[*c*]:RES:RANG[:UPP] *range* :SENS[*c*]:VOLT[:DC]:RANG[:UPP] *range* :SENS[*c*]:CURR[:DC]:RANG[:UPP]? [DEFault|MINimum|MAXimum] :SENS[*c*]:RES:RANG[:UPP]? [DEFault|MINimum|MAXimum] :SENS[*c*]:VOLT[:DC]:RANG[:UPP]? [DEFault|MINimum|MAXimum]	Specifies the expected measurement value, and sets the measurement range which provides the best resolution to measure the specified value. *range*=UP|DOWN|MINimum|MAXimum|DEFault| minimum to maximum measurement value, in A, Ω, or V. See “Measurement Ranges” on page 2-27.	1.00E-04 A, 20 kΩ, or 2 V
:SENS[*c*]:DATA? [*offset*[,*size*]]	Returns the array data which contains all data for the element specified by the :FORM:ELEM:SENS command. *offset*=CURRent|STARt|0 to maximum *size*=1 to maximum	STAR and all data
:SENS[*c*]:DATA:LAT?	Returns the latest data for the element specified by the :FORM:ELEM:SENS command.	
:SENS[*c*]:FUNC:OFF *fctn*[,*fctn*[,*fctn*]] :SENS[*c*]:FUNC:OFF?	Disables the specified measurement functions. fctn=“CURRent[:DC]”|“VOLTage[:DC]”|“RESistance”	“RES”

Command	Summary	Reset setting
:SENS[c]:FUNC:OFF:ALL	Disables all measurement functions.	
:SENS[c]:FUNC:OFF:COUN?	Returns the number of measurement functions that are disabled.	
:SENS[c]:FUNC[:ON] *fctn*[,*fctn*[,*fctn*]] :SENS[c]:FUNC[:ON]?	Enables the specified measurement functions. fctn="CURRent[:DC]"|"VOLTage[:DC]" |"RESistance"	"VOLT", "CURR"
:SENS[c]:FUNC[:ON]:ALL	Enables all measurement functions.	
:SENS[c]:FUNC[:ON]:COUN?	Returns the number of measurement functions that are enabled.	
:SENS[c]:FUNC:STAT? *fctn*	Returns if the specified measurement function is enabled or disabled. fctn="CURRent[:DC]"|"VOLTage[:DC]" |"RESistance"	
:SENS[c]:REM *mode* :SENS[c]:REM?	Enables or disables the remote sensing. Remote sensing must be enabled to use the 4-wire connection (Kelvin connection). *mode*=1|ON|0|OFF	OFF
:SENS[c]:RES:MODE *mode* :SENS[c]:RES:MODE?	Selects the resistance measurement mode. *mode*=MANual |AUTO	MAN
:SENS[c]:RES:OCOM *mode* :SENS[c]:RES:OCOM?	Enables or disables the offset-compensated resistance measurement. *mode*=1|ON|0|OFF	OFF

Command	Summary	Reset setting
:SENS[*c*]:TOUT:SIGN *output*{,output} :SENS[*c*]:TOUT:SIGN *output*{,output} :SENS[*c*]:TOUT:SIGN *output*{,output} :SENS[*c*]:TOUT:SIGN? :SENS[*c*]:TOUT:SIGN?	Selects the trigger output for the status change between the trigger layer and the acquire device action. *output*=INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	EXT1
:SENS[*c*]:TOUT[:STAT] *mode* :SENS[*c*]:TOUT[:STAT] *mode* :SENS[*c*]:TOUT[:STAT] *mode* :SENS[*c*]:TOUT[:STAT]? :SENS[*c*]:TOUT[:STAT]?	Enables or disables the trigger output for the status change between the trigger layer and the acquire device action. *mode*=1|ON|0|OFF	OFF
:SENS[*c*]:WAIT:AUTO *mode* :SENS[*c*]:WAIT:AUTO?	Enables or disables the initial wait time used for calculating the measurement wait time for the specified channel. See :SENS[*c*]:WAIT[:STAT]. *mode*=1|ON|0|OFF	ON

Command	Summary	Reset setting
:SENS[*c*]:WAIT:GAIN *gain* :SENS[*c*]:WAIT:GAIN? [DEFault|MINimum|MAXimum]	Sets the gain value used for calculating the measurement wait time for the specified channel. *gain*=MINimum|MAXimum|DEFault| 0 to 100	1
:SENS[*c*]:WAIT:OFFS *offset* :SENS[*c*]:WAIT:OFFS? [DEFault|MINimum|MAXimum]	Sets the offset value used for calculating the measurement wait time for the specified channel. *gain*=MINimum|MAXimum|DEFault| 0 to 1 seconds	0
:SENS[*c*]:WAIT[:STAT] *mode* :SENS[*c*]:WAIT[:STAT]?	Enables or disables the measurement wait time for the specified channel. The wait time is defined as the time the measurement channel cannot start measurement after the start of a DC output or the trailing edge of a pulse. *mode*=0 or OFF disables the measurement wait time. The wait time is set to 0. *mode*=1 or ON enables the measurement wait time given by the following formula. • :SENS[*c*]:WAIT:AUTO ON|1 condition: wait time = *gain* × initial wait time + *offset* • :SENS[*c*]:WAIT:AUTO OFF|0 condition: wait time = *offset* The initial wait time is automatically set by the instrument and cannot be changed.	ON

Table 2-3 OUTPut Subsystem

Command	Summary	Reset setting
:OUTP[*c*]:FILT:AUTO *mode* :OUTP[*c*]:FILT:AUTO?	Enables or disables the automatic filter function. *mode*=1|ON|0|OFF	OFF
:OUTP[*c*]:FILT[:LPAS]:FREQ *freq* :OUTP[*c*]:FILT[:LPAS]:FREQ? [DEFault|MINimum|MAXimum]	Sets the cutoff frequency of the output filter. This command setting is ignored if the automatic filter function is enabled by the :OUTP:FILT:AUTO command. *freq*=MINimum|MAXimum|DEFault| 31.830 Hz to +31.831 kHz *freq* = 1/(2 × π × *Tconst*)	MIN
:OUTP[*c*]:FILT[:LPAS][:STAT] *mode* :OUTP[*c*]:FILT[:LPAS][:STAT]?	Enables or disables the output filter. *mode*=1|ON|0|OFF	ON
:OUTP[*c*]:FILT[:LPAS]:TCON *Tconst* :OUTP[*c*]:FILT[:LPAS]:TCON? [DEFault|MINimum|MAXimum]	Sets the time constant instead of setting the cutoff frequency of the output filter. This command setting is ignored if the automatic filter function is enabled by the :OUTP[*c*]:FILT:AUTO command. *Tconst*=MINimum|MAXimum|DEFault| 5 µs to 5 ms *Tconst* = 1/(2 × π × *freq*)	MIN
:OUTP[*c*]:HCAP[:STAT] *mode* :OUTP[*c*]:HCAP[:STAT]?	Enables or disables the high capacitance mode. *mode*=1|ON|0|OFF	OFF
:OUTP[*c*]:LOW *low_state* :OUTP[*c*]:LOW?	Selects the state of the low terminal. *low_state*=FLOat|GROund	GRO
:OUTP[*c*]:OFF:AUTO *mode* :OUTP[*c*]:OFF:AUTO?	Enables or disables the automatic output off function. *mode*=1|ON|0|OFF	OFF

Command	Summary	Reset setting
:OUTP[*c*]:OFF:MODE *mode* :OUTP[*c*]:OFF:MODE?	Selects the source condition after output off. *mode*=ZERO|HIZ|NORMal	NORM
:OUTP[*c*]:ON:AUTO *mode* :OUTP[*c*]:ON:AUTO?	Enables or disables the automatic output on function. *mode*=1|ON|0|OFF	ON
:OUTP[*c*]:PROT[:STAT] *mode* :OUTP[*c*]:PROT[:STAT]?	Enables or disables the over voltage/ current protection. *mode*=1|ON|0|OFF	ON
:OUTP[*c*]:REC *index*	Recalls the channel setup. *index*=0 or 1 to recall the channel setup 0 or 1 saved by the :OUTP:SAVE command.	
:OUTP[*c*]:SAVE *index*	Saves the channel setup. The setup can be recalled by the :OUTP:REC command. *index*=0 or 1 to memorize the present channel setup as the channel setup 0 or 1.	
:OUTP[*c*][:STAT] *mode* :OUTP[*c*][:STAT]?	Enables or disables the source output. *mode*=1|ON|0|OFF	OFF

Source Output Ranges

Table 2-4 Voltage Output Range

Range value	Setting resolution		DC output voltage or pulse peak/base voltage	Maximum current [a]		Pulse width t [b]
	B2901A B2902A	B2911A B2912A		DC output	Pulsed output	
0.2 V	1 µV	0.1 µV	0 ≤ |V| ≤ 0.21 V	±3.03 A	±3.03 A with 50 µs ≤ t ≤ t_{max} ±10.5 A with 50 µs ≤ t ≤ 1 ms	
2 V	10 µV	1 µV	0 ≤ |V| ≤ 2.1 V			
20 V	100 µV	10 µV	0 ≤ |V| ≤ 6 V			
			6 V < |V| ≤ 21 V	±1.515 A	±1.515 A with 50 µs ≤ t ≤ t_{max}	
200 V	1 mV	100 µV	0 ≤ |V| ≤ 6 V	±3.03 A	±3.03 A with 50 µs ≤ t ≤ t_{max} ±10.5 A with 50 µs ≤ t ≤ 1 ms	
			6 V < |V| ≤ 21 V	±1.515 A	±1.515 A	50 µs ≤ t ≤ t_{max}
			21 V < |V| ≤ 210 V	±105 mA	±105 mA	
			0 ≤ |V| ≤ 180 V	—	±1.05 A	50 µs ≤ t ≤ 10 ms
			0 ≤ |V| ≤ 200 V	—	±1.515 A	50 µs ≤ t ≤ 2.5 ms

a. Table 2-5 shows the limitations when using Channels 1 and 2 for DC output or Pulsed output with 50 µs ≤ t ≤ t_{max} (=99.9999 ks).

b. Maximum duty cycle is 99.9999 % for the pulse with 50 µs ≤ t ≤ t_{max}, and 2.5 % for the pulse with 50 µs ≤ t ≤ 1 ms, 50 µs ≤ t ≤ 2.5 ms, or 50 µs ≤ t ≤ 10 ms.

Table 2-5 Limitations for using Channels 1 and 2

Channel 1 voltage V1	Channel 2 voltage V2	Current limit [a]
0 < |V1| ≤ 6 V	0 < |V2| ≤ 6 V	I1 + I2 ≤ 4 A
	6 V < |V2| ≤ 21 V	I1 + I2 × 1.6 ≤ 4 A
6 V < |V1| ≤ 21 V	0 < |V2| ≤ 6 V	I1 + I2 × 0.625 ≤ 2.5 A
	6 V < |V2| ≤ 21 V	I1 + I2 ≤ 2.5 A

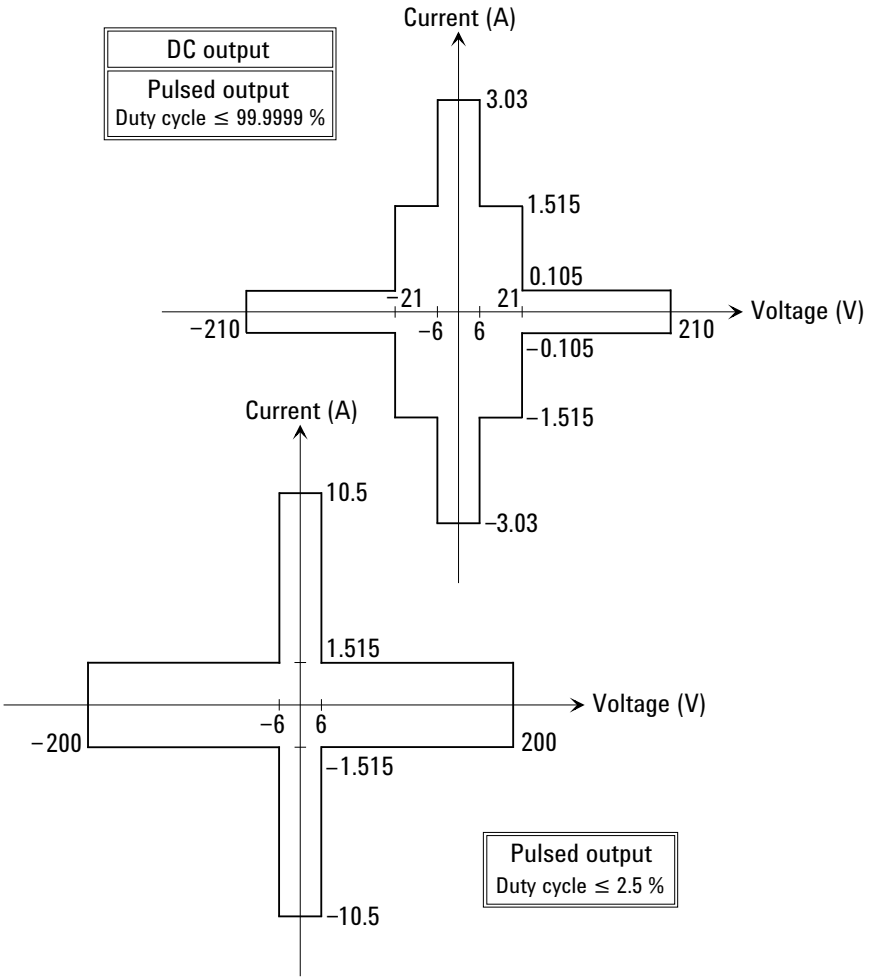
a. I1: Channel 1 current, I2: Channel 2 current

Table 2-6 Current Output Range

Range value	Setting resolution		DC output current or pulse peak/base current [a] [b]	Maximum voltage		Pulse width t [c]
	B2901A B2902A	B2911A B2912A		DC output	Pulsed output	
10 nA [d]	—	10 fA	0 ≤ |I| ≤ 10.5 nA	±210 V	±210 V	50 µs ≤ t ≤ t_{max}
100 nA	1 pA	100 fA	0 ≤ |I| ≤ 105 nA			
1 µA	10 pA	1 pA	0 ≤ |I| ≤ 1.05 µA			
10 µA	100 pA	10 pA	0 ≤ |I| ≤ 10.5 µA			
100 µA	1 nA	100 pA	0 ≤ |I| ≤ 105 µA			
1 mA	10 nA	1 nA	0 ≤ |I| ≤ 1.05 mA			
10 mA	100 nA	10 nA	0 ≤ |I| ≤ 10.5 mA			
100 mA	1 µA	100 nA	0 ≤ |I| ≤ 105 mA			
1 A	10 µA	1 µA	0 ≤ |I| ≤ 105 mA			
			105 mA < |I| ≤ 1.05 A	±21 V	±21 V	
			0 ≤ |I| ≤ 1.05 A	—	±200 V	50 µs ≤ t ≤ 2.5 ms
				—	±180 V	50 µs ≤ t ≤ 10 ms
1.5 A			0 ≤ |I| ≤ 105 mA	±210 V	±210 V	50 µs ≤ t ≤ t_{max}
			105 mA < |I| ≤ 1.515 A	±21 V	±21 V	
			0 ≤ |I| ≤ 1.515 A	—	±200 V	50 µs ≤ t ≤ 2.5 ms
			0 ≤ |I| ≤ 1.05 A	—	±180 V	50 µs ≤ t ≤ 10 ms
3 A	100 µA	10 µA	0 ≤ |I| ≤ 105 mA	±210 V	±210 V	50 µs ≤ t ≤ t_{max}
			105 mA < |I| ≤ 1.515 A	±21 V	±21 V	
			1.515 A < |I| ≤ 3.03 A	±6 V	±6 V	
10 A			0 ≤ |I| ≤ 10.5 A	—	±6 V	50 µs ≤ t ≤ 1 ms
			0 ≤ |I| ≤ 1.515 A	—	±200 V	50 µs ≤ t ≤ 2.5 ms
			0 ≤ |I| ≤ 1.05 A	—	±180 V	50 µs ≤ t ≤ 10 ms

a. Table 2-5 shows the limitations when using Channels 1 and 2 for DC output or Pulsed output with 50 µs ≤ t ≤ t_{max} (=99.9999 ks).
b. Maximum base current is 500 mA for the pulse with 50 µs ≤ t ≤ 1 ms, and 50 ms for the pulse with 50 µs ≤ t ≤ 2.5 ms or 50 µs ≤ t ≤ 10 ms.
c. Maximum duty cycle is 99.9999 % for the pulse with 50 µs ≤ t ≤ t_{max}, and 2.5 % for the pulse with 50 µs ≤ t ≤ 1 ms, 50 µs ≤ t ≤ 2.5 ms, or 50 µs ≤ t ≤ 10 ms.
d. Available for the B2911A and B2912A. Not available for the B2901A and B2902A.

Figure 2-5 Maximum Voltage and Current



Measurement Ranges

Table 2-7 **Voltage Measurement Range**

Range value	Voltage measurement value	Resolution
0.2 V	0 ≤ |V| ≤ 0.212 V	0.1 µV
2 V	0 ≤ |V| ≤ 2.12 V	1 µV
20 V	0 ≤ |V| ≤ 21.2 V	10 µV
200 V	0 ≤ |V| ≤ 212 V	100 µV

Table 2-8 **Current Measurement Range**

Range value	Current measurement value	Resolution
10 nA [a]	0 ≤ |I| ≤ 10.6 nA	10 fA
100 nA	0 ≤ |I| ≤ 106 nA	100 fA
1 µA	0 ≤ |I| ≤ 1.06 µA	1 pA
10 µA	0 ≤ |I| ≤ 10.6 µA	10 pA
100 µA	0 ≤ |I| ≤ 106 µA	100 pA
1 mA	0 ≤ |I| ≤ 1.06 mA	1 nA
10 mA	0 ≤ |I| ≤ 10.6 mA	10 nA
100 mA	0 ≤ |I| ≤ 106 mA	100 nA
1 A	0 ≤ |I| ≤ 1.06 A	1 µA
1.5 A	0 ≤ |I| ≤ 1.53 A	
3 A	0 ≤ |I| ≤ 3.06 A	10 µA
10 A [b]	0 ≤ |I| ≤ 10.6 A	

a. Available for the B2911A and B2912A. Not available for the B2901A and B2902A.
b. Available for pulse mode. Not available for DC mode.

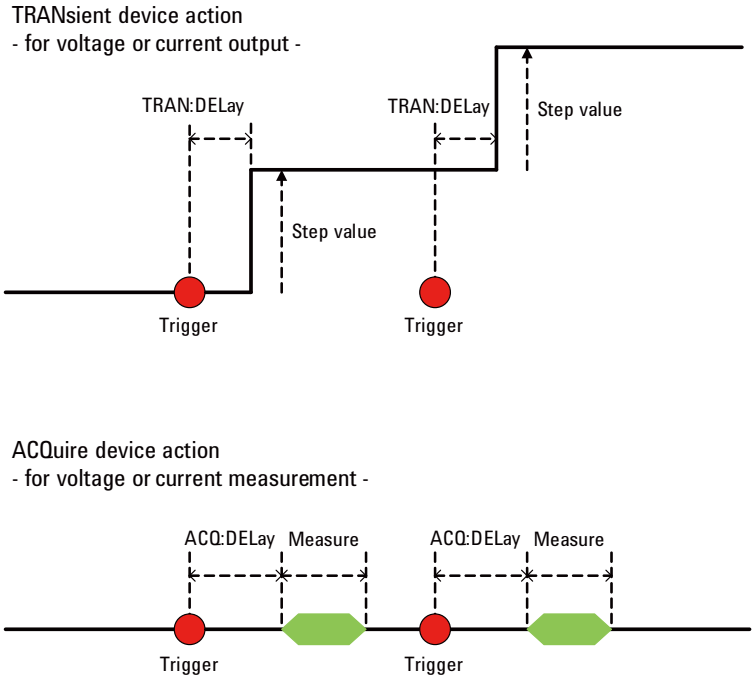
Table 2-9 **Resistance Measurement Range[1]**

Range value	Resistance measurement value	Display resolution	Test current
2 Ω	0 < R ≤ 2 Ω	1 μΩ	1A
20 Ω	2 Ω < R ≤ 20 Ω	10 μΩ	100 mA
200 Ω	20 Ω < R ≤ 200 Ω	100 μΩ	10 mA
2 kΩ	200 Ω < R ≤ 2 kΩ	1 mΩ	1 mA
20 kΩ	2 kΩ < R ≤ 20 kΩ	10 mΩ	100 μA
200 kΩ	20 kΩ < R ≤ 200 kΩ	100 mΩ	10 μA
2 MΩ	200 kΩ < R ≤ 2 MΩ	1 Ω	1 μA
20 MΩ	2 MΩ < R ≤ 20 MΩ	10 Ω	100 nA
200 MΩ	20 MΩ < R ≤ 200 MΩ	100 Ω	10 nA

1. The resistance measurement range is effective for the resistance measurements set to the AUTO mode which is selected by the :SENSe:RESistance:MODE command.

Controlling Source/Measure Trigger

Figure 2-6 **Transient and Acquire Device Actions**



NOTE If channels are set as shown below, the device actions start simultaneously.

- To synchronize transient actions (source output)
  - Trigger source is set to the same mode.
  - Delay time is set to the same value.
  - Source output ranging mode is set to the fixed mode.
  - Source wait time control is set to OFF.
  - Measurement wait time control is set to OFF.
  - Measurement ranging mode is set to the fixed mode.

- To synchronize acquire actions (measurement)
  - Trigger source is set to the same mode.
  - Delay time is set to the same value.
  - Measurement wait time control is set to OFF.
  - Measurement ranging mode is set to the fixed mode.

Figure 2-7 Operation Example Using Trigger Delay and AINT Trigger Source

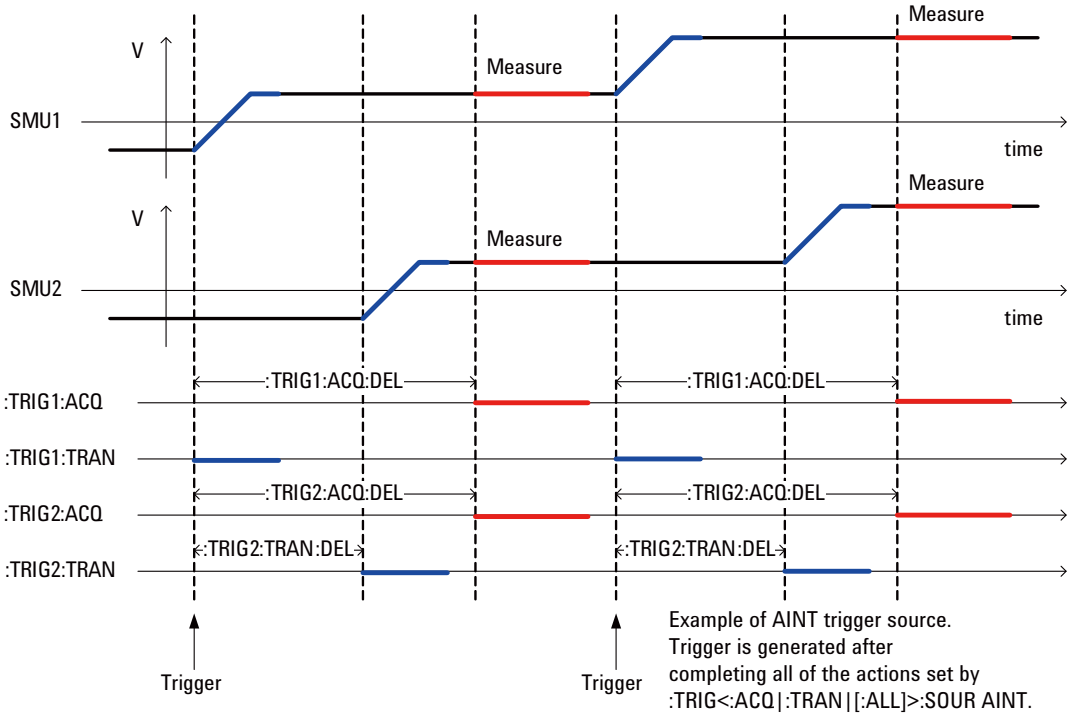


Figure 2-8 B2900 Trigger System

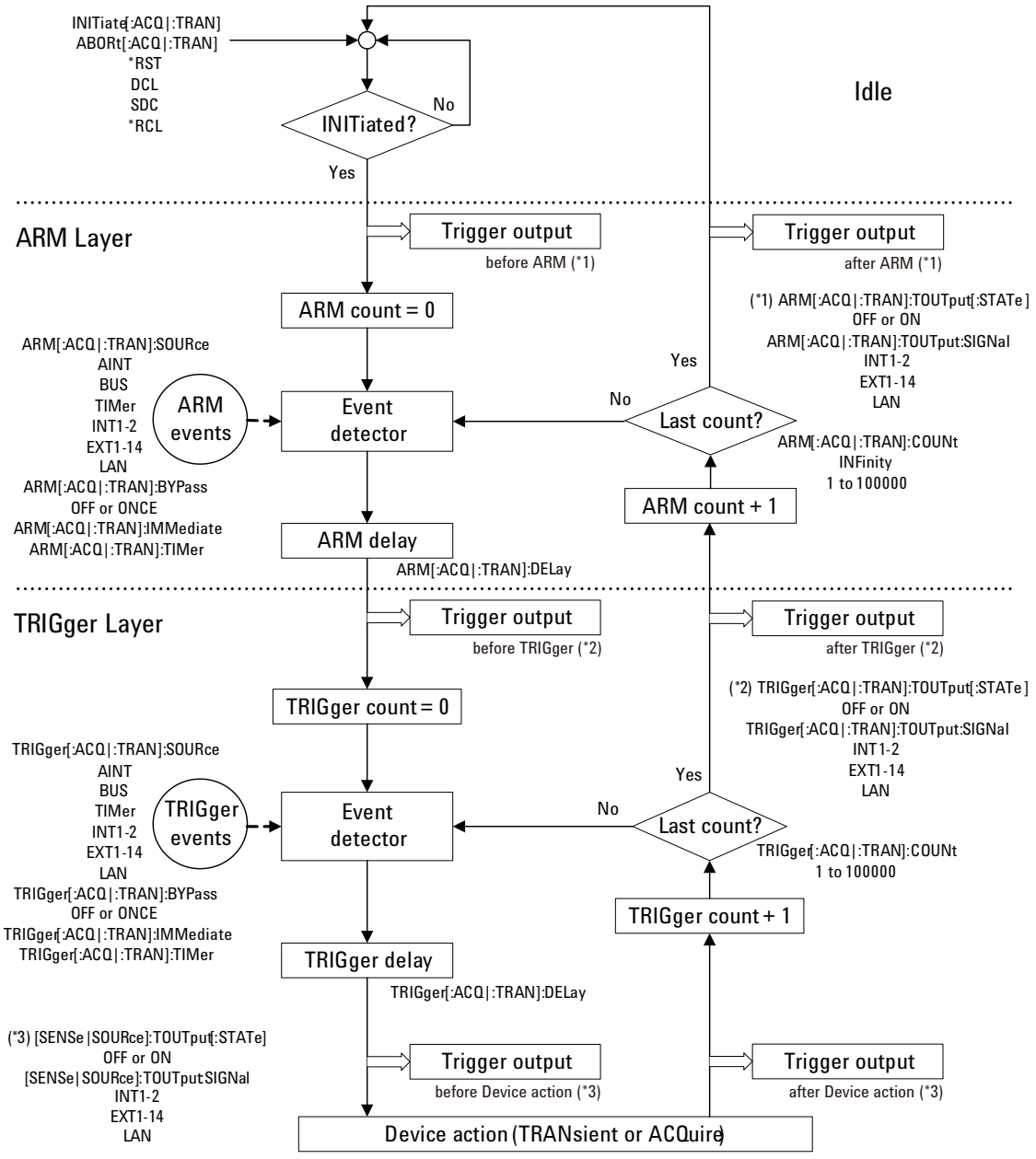


Table 2-10 TRIGger Subsystem

Command	Summary	Reset setting
:ABOR:ACQ [*chanlist*] :ABOR:TRAN [*chanlist*] :ABOR[:ALL] [*chanlist*]	Aborts the specified device action for the specified channel. Trigger status is changed to idle. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	(@1) for 1-ch models (@1,2) for 2-ch models
:ARM:ACQ[:IMM] [*chanlist*] :ARM:TRAN[:IMM] [*chanlist*] :ARM[:ALL][:IMM] [*chanlist*]	Sends an immediate arm trigger for the specified device action to the specified channel. When the status of the specified device action is initiated, the arm trigger causes a layer change from arm to trigger. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	(@1) for 1-ch models (@1,2) for 2-ch models
:ARM[*c*]:ACQ[:LAY]:BYP *bypass* :ARM[*c*]:TRAN[:LAY]:BYP *bypass* :ARM[*c*][:ALL][:LAY]:BYP *bypass* :ARM[*c*]:ACQ[:LAY]:BYP? :ARM[*c*]:TRAN[:LAY]:BYP?	Enables or disables a bypass for the event detector in the arm layer. *bypass*=ONCE|OFF ONCE enables the bypass only for the first passage.	OFF
:ARM[*c*]:ACQ[:LAY]:COUN *count* :ARM[*c*]:TRAN[:LAY]:COUN *count* :ARM[*c*][:ALL][:LAY]:COUN *count* :ARM[*c*]:ACQ[:LAY]:COUN? [*count*] :ARM[*c*]:TRAN[:LAY]:COUN? [*count*] :ARM[*c*][:ALL][:LAY]:COUN? *count*	Sets the arm count for the specified device action. *count*=INFinity|MINimum|MAXimum|DEFault| 1 to 100000 or 2147483647 *count*=2147483647 indicates infinity. Query does not support *count*=INFinity, 1 to 100000 and 2147483647. *Arm count* × *Trigger count* must be less than 100001.	1

Command	Summary	Reset setting
:ARM[*c*]:ACQ[:LAY]:DEL *delay* :ARM[*c*]:TRAN[:LAY]:DEL *delay* :ARM[*c*][:ALL][:LAY]:DEL *delay* :ARM[*c*]:ACQ[:LAY]:DEL? [*delay*] :ARM[*c*]:TRAN[:LAY]:DEL? [*delay*] :ARM[*c*][:ALL][:LAY]:DEL? *delay*	Sets the arm delay for the specified device action. *delay*=MINimum|MAXimum|DEFault|0 to 100 seconds Query does not support *delay*=0 to 100.	0
:ARM[*c*]:ACQ[:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :ARM[*c*]:TRAN[:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :ARM[*c*][:ALL][:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :ARM[*c*]:ACQ[:LAY]:SOUR:LAN? :ARM[*c*]:TRAN[:LAY]:SOUR:LAN?	Specifies one or more LXI triggers used for the arm source for the specified device action. *lan_id*=LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7	All is selected.
:ARM[*c*]:ACQ[:LAY]:SOUR[:SIGN] *source* :ARM[*c*]:TRAN[:LAY]:SOUR[:SIGN] *source* :ARM[*c*][:ALL][:LAY]:SOUR[:SIGN] *source* :ARM[*c*]:ACQ[:LAY]:SOUR[:SIGN]? :ARM[*c*]:TRAN[:LAY]:SOUR[:SIGN]?	Selects the arm source for the specified device action. *source*=AINT|BUS|TIMer|INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	AINT
:ARM[*c*]:ACQ[:LAY]:TIM *interval* :ARM[*c*]:TRAN[:LAY]:TIM *interval* :ARM[*c*][:ALL][:LAY]:TIM *interval* :ARM[*c*]:ACQ[:LAY]:TIM? [*interval*] :ARM[*c*]:TRAN[:LAY]:TIM? [*interval*] :ARM[*c*][:ALL][:LAY]:TIM? *interval*	Sets the interval of the TIMer arm source for the specified device action. *interval*=MINimum|MAXimum|DEFault|1E-5 to 1E+5 seconds Query does not support *interval*=1E-5 to 1E+5.	1E-5 seconds

Command	Summary	Reset setting
:ARM[*c*]:ACQ[:LAY]:TOUT:SIGN *output*{,output} :ARM[*c*]:TRAN[:LAY]:TOUT:SIGN *output*{,output} :ARM[*c*][:ALL][:LAY]:TOUT:SIGN *output*{,output} :ARM[*c*]:ACQ[:LAY]:TOUT:SIGN? :ARM[*c*]:TRAN[:LAY]:TOUT:SIGN?	Selects the trigger output for the status change between the idle state and the arm layer. *output*=INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	EXT1
:ARM[*c*]:ACQ[:LAY]:TOUT[:STAT] *mode* :ARM[*c*]:TRAN[:LAY]:TOUT[:STAT] *mode* :ARM[*c*][:ALL][:LAY]:TOUT[:STAT] *mode* :ARM[*c*]:ACQ[:LAY]:TOUT[:STAT]? :ARM[*c*]:TRAN[:LAY]:TOUT[:STAT]?	Enables or disables the trigger output for the status change between the idle state and the arm layer. *mode*=1|ON|0|OFF	OFF
:IDLE[*c*]:ACQ? :IDLE[*c*]:TRAN? :IDLE[*c*][:ALL]?	Checks the status of the specified device action for the specified channel, and waits until the status is changed to idle.	
:INIT[:IMM]:ACQ [*chanlist*] :INIT[:IMM]:TRAN [*chanlist*] :INIT[:IMM][:ALL] [*chanlist*]	Initiates the specified device action for the specified channel. Trigger status is changed from idle to initiated. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:TRIG[*c*]:ACQ:BYP *bypass* :TRIG[*c*]:TRAN:BYP *bypass* :TRIG[*c*][:ALL]:BYP *bypass* :TRIG[*c*]:ACQ:BYP? :TRIG[*c*]:TRAN:BYP?	Enables or disables a bypass for the event detector in the trigger layer. *bypass*=ONCE|OFF ONCE enables the bypass only for the first passage.	OFF

Command	Summary	Reset setting
:TRIG[*c*]:ACQ:COUN *count* :TRIG[*c*]:TRAN:COUN *count* :TRIG[*c*][:ALL]:COUN *count* :TRIG[*c*]:ACQ:COUN? [*count*] :TRIG[*c*]:TRAN:COUN? [*count*] :TRIG[*c*][:ALL]:COUN? *count*	Sets the trigger count for the specified device action. *count*=MINimum|MAXimum|DEFault|1 to 100000 Query does not support *count*=1 to 100000. *Arm count* × *Trigger count* must be less than 100001.	1
:TRIG[*c*]:ACQ:DEL *delay* :TRIG[*c*]:TRAN:DEL *delay* :TRIG[*c*][:ALL]:DEL *delay* :TRIG[*c*]:ACQ:DEL? [*delay*] :TRIG[*c*]:TRAN:DEL? [*delay*] :TRIG[*c*][:ALL]:DEL? *delay*	Sets the trigger delay for the specified device action. *delay*=MINimum|MAXimum|DEFault|0 to 100 seconds Query does not support *delay*=0 to 100.	0
:TRIG:ACQ[:IMM] [*chanlist*] :TRIG:TRAN[:IMM] [*chanlist*] :TRIG[:ALL][:IMM] [*chanlist*]	Sends an immediate trigger for the specified device action to the specified channel. When the status of the specified device action is initiated, the trigger causes the specified device action. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	(@1) for 1-ch models (@1,2) for 2-ch models
:TRIG[*c*]:ACQ[:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :TRIG[*c*]:TRAN[:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :TRIG[*c*][:ALL][:LAY]:SOUR:LAN *lan_id*{,*lan_id*} :TRIG[*c*]:ACQ[:LAY]:SOUR:LAN? :TRIG[*c*]:TRAN[:LAY]:SOUR:LAN?	Specifies one or more LXI triggers used for the trigger source for the specified device action. *lan_id*=LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7	All is selected.

Command	Summary	Reset setting
:TRIG[*c*]:ACQ:SOUR[:SIGN] *source* :TRIG[*c*]:TRAN:SOUR[:SIGN] *source* :TRIG[*c*][:ALL]:SOUR[:SIGN] *source* :TRIG[*c*]:ACQ:SOUR[:SIGN]? :TRIG[*c*]:TRAN:SOUR[:SIGN]?	Selects the trigger source for the specified device action. *source*=AINT|BUS|TIMer|INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	AINT
:TRIG[*c*]:ACQ:TIM *interval* :TRIG[*c*]:TRAN:TIM *interval* :TRIG[*c*][:ALL]:TIM *interval* :TRIG[*c*]:ACQ:TIM? [*interval*] :TRIG[*c*]:TRAN:TIM? [*interval*] :TRIG[*c*][:ALL]:TIM? *interval*	Sets the interval of the TIMer trigger source for the specified device action. *interval*=MINimum|MAXimum|DEFault|1E-5 to 1E+5 seconds Query does not support *interval*=1E-5 to 1E+5.	1E-5 seconds
:TRIG[*c*]:ACQ:TOUT:SIGN *output*{,output} :TRIG[*c*]:TRAN:TOUT:SIGN *output*{,output} :TRIG[*c*][:ALL]:TOUT:SIGN *output*{,output} :TRIG[*c*]:ACQ:TOUT:SIGN? :TRIG[*c*]:TRAN:TOUT:SIGN?	Selects the trigger output for the status change between the arm layer and the trigger layer. *output*=INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14	EXT1
:TRIG[*c*]:ACQ:TOUT[:STAT] *mode* :TRIG[*c*]:TRAN:TOUT[:STAT] *mode* :TRIG[*c*][:ALL]:TOUT[:STAT] *mode* :TRIG[*c*]:ACQ:TOUT[:STAT]? :TRIG[*c*]:TRAN:TOUT[:STAT]?	Enables or disables the trigger output for the status change between the arm layer and the trigger layer. *mode*=1|ON|0|OFF	OFF

Reading Source/Measure Data

Figure 2-9 **Measurement Data Flow**

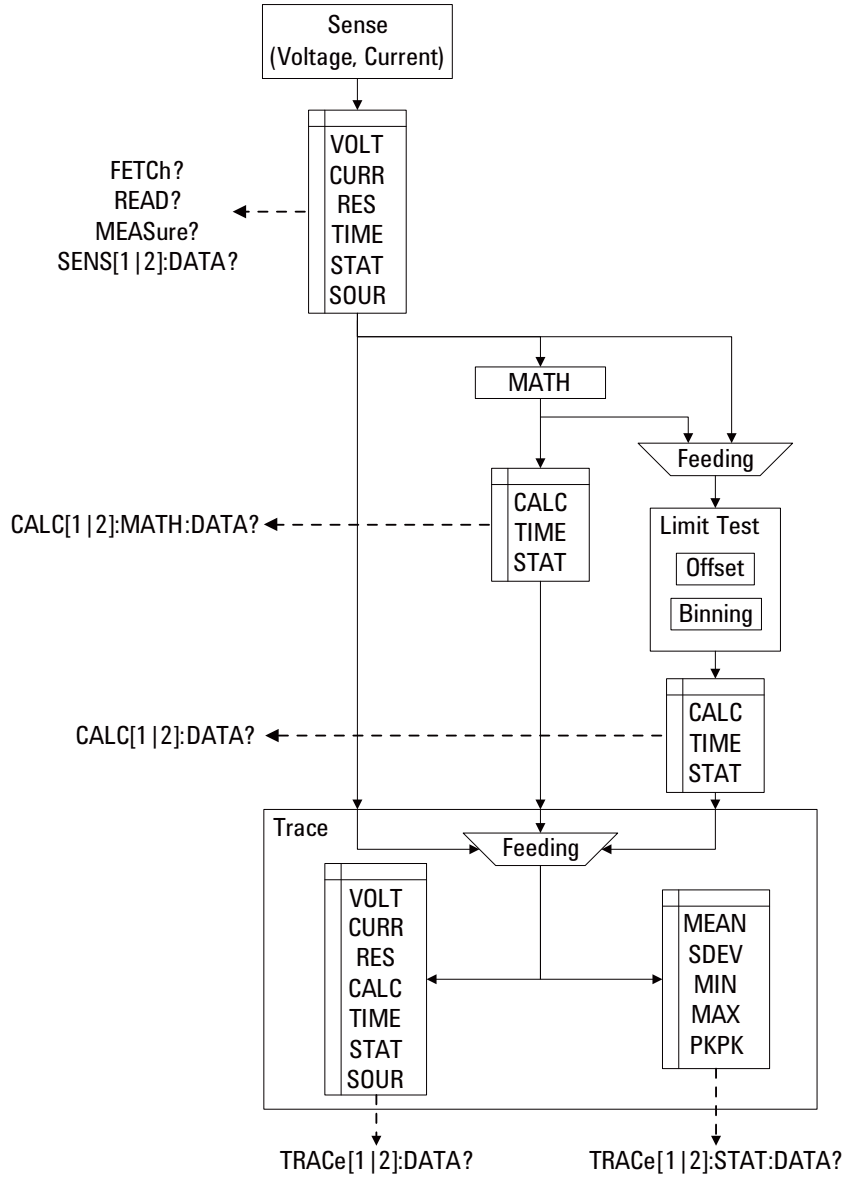


Figure 2-10 **Composite Limit Test Flowchart Example for Sorting Mode**

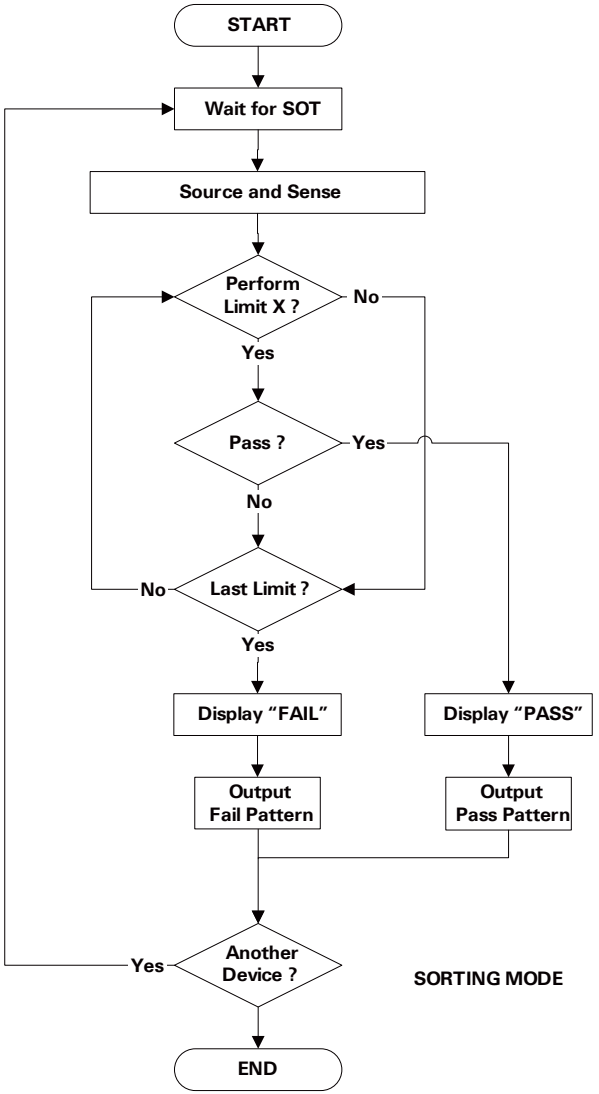


Figure 2-11 **Composite Limit Test Flowchart Example for Grading Mode**

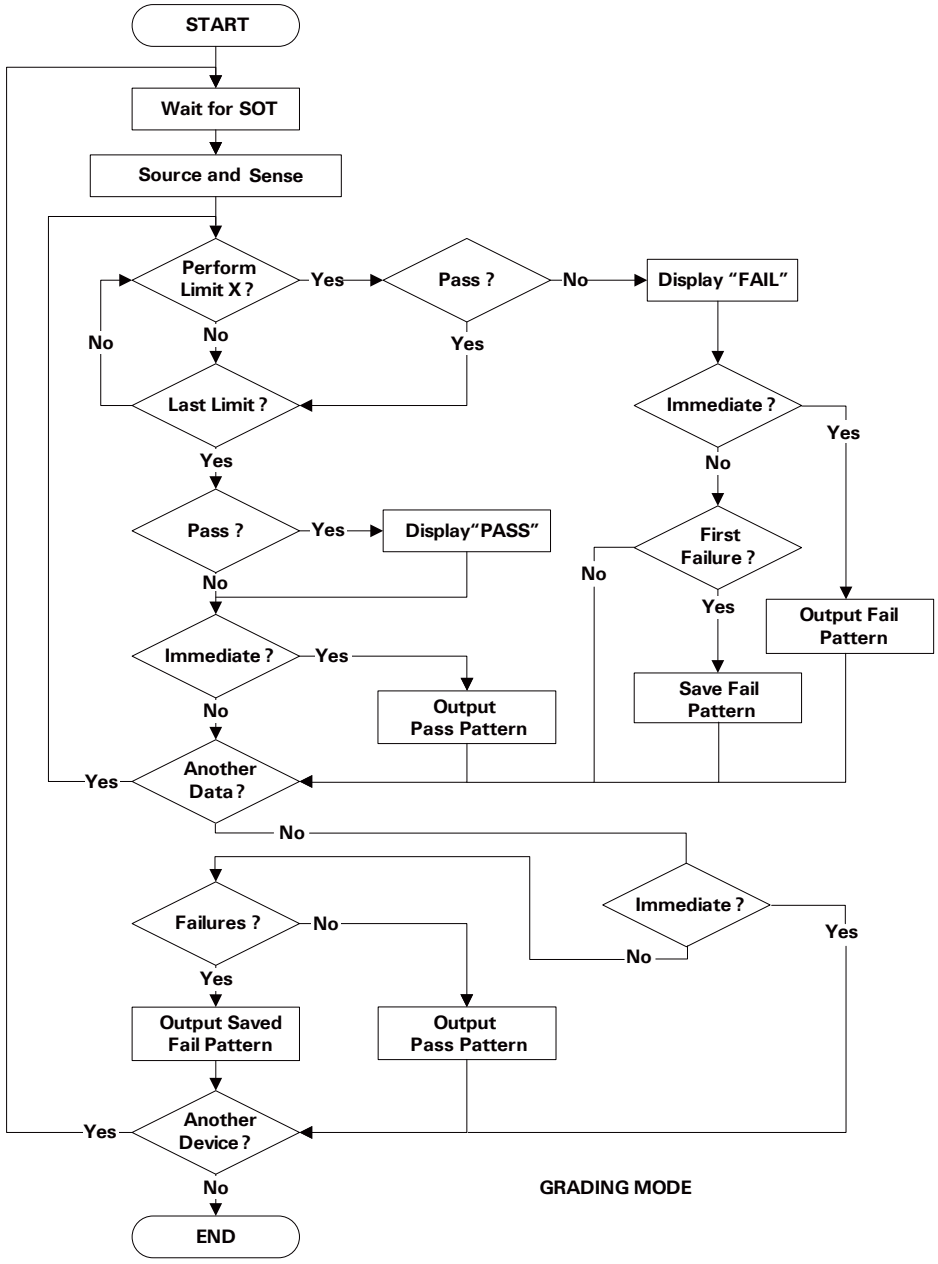


Table 2-11 FETCh Subsystem

Command	Summary	Reset setting
:FETC:ARR? [*chanlist*]	Returns the array data which contains all of the voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORM:ELEM:SENS command. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:FETC:ARR:CURR? [*chanlist*] :FETC:ARR:RES? [*chanlist*] :FETC:ARR:SOUR? [*chanlist*] :FETC:ARR:STAT? [*chanlist*] :FETC:ARR:TIME? [*chanlist*] :FETC:ARR:VOLT? [*chanlist*]	Returns the array data which contains all of the current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURR, RES, SOUR, STAT, TIME, or VOLT. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:FETC[:SCAL]? [*chanlist*]	Returns the latest voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORM:ELEM:SENS command. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:FETC[:SCAL]:CURR? [*chanlist*] :FETC[:SCAL]:RES? [*chanlist*] :FETC[:SCAL]:SOUR? [*chanlist*] :FETC[:SCAL]:STAT? [*chanlist*] :FETC[:SCAL]:TIME? [*chanlist*] :FETC[:SCAL]:VOLT? [*chanlist*]	Returns the latest current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURR, RES, SOUR, STAT, TIME, or VOLT. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	

Table 2-12 FORMat Subsystem

Command	Summary	Reset setting
`:FORM:BORD` *byte_order* `:FORM:BORD?`	Sets the byte order of binary output data. *byte_order*=NORMal|SWAPped	NORM
`:FORM[:DATA]` *format* `:FORM[:DATA]?`	Sets the data output format. *format*=ASCii|REAL,32|REAL,64	ASC
`:FORM:DIG` *format* `:FORM:DIG?`	Sets the response format of the bit pattern defined by :CALC:xxxx:DIG[:DATA]. *format*=ASCii|BINary|OCTal|HEXadecimal	ASC
`:FORM:ELEM:CALC` *type*`{,`*type*`}` `:FORM:ELEM:CALC?`	Specifies the elements included in the calculation result data. *type*=CALC|TIME|STATus Order of returned data: *calc*, *time*, *status*	CALC
`:FORM:ELEM:SENS` *type*`{,`*type*`}` `:FORM:ELEM:SENS?`	Specifies the elements included in the sense or measurement result data. *type*=VOLTage|CURRent|RESistance|TIME|STATus|SOURce Order of returned data: *voltage*, *current*, *resistance*, *time*, *status*, *source*	VOLT, CURR, RES, TIME, STAT, SOUR
`:FORM:SREG` *format* `:FORM:SREG?`	Sets the response format of the status byte register. *format*=ASCii|BINary|OCTal|HEXadecimal	ASC

Table 2-13 READ Subsystem

Command	Summary	Reset setting
:READ:ARR? [*chanlist*]	Executes the :INIT command and the :FETC:ARR? command in series, and returns the array data which contains all data for the element specified by the :FORM:ELEM:SENS command. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:READ:ARR:CURR? [*chanlist*] :READ:ARR:RES? [*chanlist*] :READ:ARR:SOUR? [*chanlist*] :READ:ARR:STAT? [*chanlist*] :READ:ARR:TIME? [*chanlist*] :READ:ARR:VOLT? [*chanlist*]	Executes the :INIT command and the :FETC:ARR:<CURR|RES|SOUR|STAT|TIME|VOLT>? command in series, and returns the array data which contains all data for the element specified by CURR, RES, SOUR, STAT, TIME, or VOLT. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:READ[:SCAL]? [*chanlist*]	Executes the :INIT command and the :FETC? command in series, and returns the latest data for the element specified by the :FORM:ELEM:SENS command. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:READ[:SCAL]:CURR? [*chanlist*] :READ[:SCAL]:RES? [*chanlist*] :READ[:SCAL]:SOUR? [*chanlist*] :READ[:SCAL]:STAT? [*chanlist*] :READ[:SCAL]:TIME? [*chanlist*] :READ[:SCAL]:VOLT? [*chanlist*]	Executes the :INIT command and the :FETC:<CURR|RES|SOUR|STAT|TIME|VOLT>? command in series, and returns the latest data for the element specified by CURR, RES, SOUR, STAT, TIME, or VOLT. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	

Table 2-14 MEASure Subsystem

Command	Summary	Reset setting
:MEAS? [*chanlist*]	Executes a spot measurement (one-shot measurement) and returns the measurement result data. Measurement items can be specified by the :FORM:ELEM:SENS command. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:MEAS:CURR[:DC]? [*chanlist*] :MEAS:RES? [*chanlist*] :MEAS:VOLT[:DC]? [*chanlist*]	Executes a spot measurement (one-shot measurement) and returns the measurement result data. Measurement items can be specified by CURR, RES, or VOLT. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	

Table 2-15 CALCulate Subsystem

Command	Summary	Reset setting
:CALC[*c*]:CLIM:CLE:AUTO *mode* :CALC[*c*]:CLIM:CLE:AUTO?	Enables or disables the automatic clear function of the composite limit test. *mode*=0|OFF|1|ON	ON
:CALC[*c*]:CLIM:CLE:AUTO:DEL *time* :CALC[*c*]:CLIM:CLE:AUTO:DEL?	Sets the delay time for the automatic clear of the composite limit test. *time*=MINimum|MAXimum|DEFault|1E-5 to 60 seconds Query does not support *time*=1E-5 to 60.	1E-4
:CALC[*c*]:CLIM:CLE[:IMM]	Clears the composite limit test results and the GPIO lines immediately.	

Command	Summary	Reset setting
:CALC[*c*]:CLIM:FAIL:DIG[:DATA] *pattern* :CALC[*c*]:CLIM:FAIL:DIG[:DATA]?	Defines a fail pattern that appears near the end of the flowcharts shown in Figures 2-10 and 2-11. This is a bit pattern used to indicate the composite limit test *fail*. *pattern*=0 to 16383	0
:CALC[*c*]:CLIM:MODE *mode* :CALC[*c*]:CLIM:MODE?	Sets the operation mode of the composite limit test. *mode*=GRADing|SORTing.	GRAD
:CALC[*c*]:CLIM:PASS:DIG[:DATA] *pattern* :CALC[*c*]:CLIM:PASS:DIG[:DATA]?	Defines a pass pattern that appears near the end of the flowcharts shown in Figures 2-10 and 2-11. This is a bit pattern used to indicate the composite limit test *pass*. *pattern*=0 to 16383	0
:CALC[*c*]:CLIM:STAT *mode* :CALC[*c*]:CLIM:STAT?	Enables or disables the composite limit test. *mode*=0|OFF|1|ON	ON
:CALC[*c*]:CLIM:STAT:ANY?	Checks if the present composite limit test contains a limit test, which is a pass/fail judgement. It is performed at the “Pass?” step in Figures 2-10 and 2-11.	
:CALC[*c*]:CLIM:UPD *result* :CALC[*c*]:CLIM:UPD?	Only for the GRAD composite limit test. Enables or disables the immediate result output or update. See “Immediate?” shown in Figure 2-11. *result*=END|IMMediate	IMM

Command	Summary	Reset setting
:CALC[*c*]:DATA? [*offset*[,*size*]]	:CALC[*c*]:DATA? returns limit test data. *offset*=CURRent|STARt|0 to maximum *size*=1 to maximum :CALC[*c*]:DATA:LAT? returns the last limit test data. Elements of the returned data are specified by the :FORM:ELEM:CALC command. The limit test data can be expressed by the following formula. *limit test data = input data - null offset*	STAR and all data
:CALC[*c*]:DATA:LAT?		
:CALC[*c*]:DIG:BIT *pin* :CALC[*c*]:DIG:BIT?	Assigns the GPIO pins used for the result output. The output is the pass/fail bit pattern. *pin*=EXT*n*|NONE. *n*=1 to 14.	NONE
:CALC[*c*]:DIG:BUSY *pin* :CALC[*c*]:DIG:BUSY?	Assigns the GPIO pin for the BUSY (busy) line for the composite limit test. *pin*=EXT*n*|NONE. *n*=1 to 14.	NONE
:CALC[*c*]:DIG:EOT *pin* :CALC[*c*]:DIG:EOT?	Assigns the GPIO pin for the EOT (end of test) line for the composite limit test. *pin*=EXT*n*|NONE. *n*=1 to 14.	NONE
:CALC[*c*]:DIG:SOT *pin* :CALC[*c*]:DIG:SOT?	Assigns the GPIO pin for the SOT (start of test) line for the composite limit test. *pin*=EXT*n*|NONE. *n*=1 to 14.	NONE
:CALC[*c*]:FEED *type* :CALC[*c*]:FEED?	Specifies the *input data* value used for calculating the limit test data. The limit test is a pass/fail judgement performed during a composite limit test. The limit test is performed at the “Pass?” step in Figures 2-10 and 2-11. *type*=MATH|RESistance|CURRent|VOLTage	VOLT

Command	Summary	Reset setting
:CALC[*c*]:LIM[*m*]:COMP:DIG[:DATA] *pattern* :CALC[*c*]:LIM[*m*]:COMP:DIG[:DATA]?	Defines the bit pattern used to indicate a *failure* of the compliance status check specified by *m*. *pattern*=0 to 16383	0
:CALC[*c*]:LIM[*m*]:COMP:FAIL *criteria* :CALC[*c*]:LIM[*m*]:COMP:FAIL?	Sets the judgement criteria for the compliance status check specified by *m*. *criteria*=IN determines that the limit test has failed if the channel goes into the compliance state. *criteria*=OUT determines that the limit test has failed if the channel comes out of the compliance state.	IN
:CALC[*c*]:LIM[*m*]:FAIL?	Returns the result of the limit test specified by *m*. 0: Passed, 1: Failed.	
:CALC[*c*]:LIM[*m*]:FUNC *type* :CALC[*c*]:LIM[*m*]:FUNC?	Sets the type of the limit test specified by *m*. *type*=COMP sets the compliance status check which checks if the channel is in the compliance status. *type*=LIM sets the limit test which checks if the measurement value is between the upper limit and the lower limit.	LIM
:CALC[*c*]:LIM[*m*]:LOW *limit* :CALC[*c*]:LIM[*m*]:LOW? [MINimum|MAXimum|DEFault]	Sets a lower limit used for the limit test specified by *m*. *limit*=MINimum|MAXimum|DEFault| -9.999999E+20 to +9.999999E+20	-1
:CALC[*c*]:LIM[*m*]:LOW:DIG[:DATA] *pattern* :CALC[*c*]:LIM[*m*]:LOW:DIG[:DATA]?	Defines the bit pattern used to indicate *failed-by-exceeding-lower-limit* of the limit test specified by *m*. The bit pattern is used for the GRAD composite limit test. *pattern*=0 to 16383	0

Command	Summary	Reset setting
:CALC[*c*]:LIM[*m*]:PASS:DIG[:DATA] *pattern* :CALC[*c*]:LIM[*m*]:PASS:DIG[:DATA]?	Defines the bit pattern used to indicate a *pass* of the limit test specified by *m*. The bit pattern is used for the SORT composite limit test. *pattern*=0 to 16383	0
:CALC[*c*]:LIM[*m*]:STAT *mode* :CALC[*c*]:LIM[*m*]:STAT?	Enables or disables the limit test specified by *m*. *mode*=1|ON|0|OFF	OFF
:CALC[*c*]:LIM[*m*]:UPP *limit* :CALC[*c*]:LIM[*m*]:UPP? [MINimum|MAXimum|DEFault]	Sets a upper limit used for the limit test specified by *m*. *limit*=MINimum|MAXimum|DEFault| -9.999999E+20 to +9.999999E+20	1
:CALC[*c*]:LIM[*m*]:UPP:DIG[:DATA] *pattern* :CALC[*c*]:LIM[*m*]:UPP:DIG[:DATA]?	Defines a bit pattern used to indicate *failed-by-exceeding-upper-limit* of the limit test specified by *m*. The bit pattern is used for the GRAD composite limit test. *pattern*=0 to 16383	0
:CALC[*c*]:MATH:DATA? [*offset*[,*size*]]	:CALC[*c*]:MATH:DATA? returns the calculation result data. *offset*=CURRent|STARt|0 to maximum *size*=1 to maximum :CALC[*c*]:MATH:DATA:LAT? returns the latest calculation result data. Elements of the returned data are specified by the :FORM:ELEM:CALC command. Math expression for the calculation is specified by the :CALC:MATH[:EXPR]:NAME and :CALC:MATH[:EXPR][:DEF] commands.	STAT and all data
:CALC[*c*]:MATH:DATA:LAT?		

Command	Summary	Reset setting
:CALC[*c*]:MATH[:EXPR]:CAT?	Returns the list of all the predefined and user-defined math expression names.	POWER, OFFCOMPOHM, VOLTCOEF, VARALPHA
:CALC[*c*]:MATH[:EXPR][:DEF] *definition*	Defines a math expression which will be a user-defined math expression. Maximum of 32 math expressions can be defined including the predefined math expressions. *definition*: Up to 256 ASCII characters.	(VOLT * CURR)
:CALC[*c*]:MATH[:EXPR]:DEL:ALL	Deletes all user-defined math expressions.	
:CALC[*c*]:MATH[:EXPR]:DEL[:SEL] *name*	Deletes an user-defined math expression. *name*: Up to 32 ASCII characters.	
:CALC[*c*]:MATH[:EXPR]:NAME *name* :CALC[*c*]:MATH[:EXPR]:NAME?	Selects a math expression used for calculation. *name*: Up to 32 ASCII characters without any control characters, space characters, single and double quotes, and comma.	"POWER"
:CALC[*c*]:MATH:STAT *mode* :CALC[*c*]:MATH:STAT?	Enables or disables the math expression. *mode*=1|ON|0|OFF	OFF
:CALC[*c*]:MATH:UNIT *name* :CALC[*c*]:MATH:UNIT?	Defines the unit name for the math expression. *name*: Up to 32 ASCII characters.	"W"

Command	Summary	Reset setting
:CALC[*c*]:OFFS *offset* :CALC[*c*]:OFFS? [MINimum|MAXimum|DEFault]	Sets the *null offset* value used for calculating the limit test data. *offset*=MINimum|MAXimum|DEFault| -9.999999E+20 to +9.999999E+20	0
:CALC[*c*]:OFFS:ACQ	Automatically sets the *null offset* value used for calculating the limit test data.	
:CALC[*c*]:OFFS:STAT *mode* :CALC[*c*]:OFFS:STAT?	Enables or disables the null offset function used for calculating the limit test data. *mode*=1|ON|0|OFF	OFF

Table 2-16 TRACe Subsystem

Command	Summary	Reset setting
:TRAC[*c*]:CLE	Clears the trace buffer of the specified channel. This command is effective when the trace buffer control mode is set to NEV by the :TRAC[*c*]:FEED:CONT command.	
:TRAC[*c*]:DATA? [*offset*[,*size*]]	Returns the data in the trace buffer. *offset*=CURRent|STARt|0 to maximum *size*=1 to maximum	
:TRAC[*c*]:FEED *type* :TRAC[*c*]:FEED?	Specifies the data placed in the trace buffer. This command is effective when the trace buffer control mode is set to NEV by the :TRAC[*c*]:FEED:CONT command. *type*=MATH|LIMit|SENSe	SENS
:TRAC[*c*]:FEED:CONT *mode* :TRAC[*c*]:FEED:CONT?	Selects the trace buffer control. *mode*=NEXT|NEVer	NEV

Command	Summary	Reset setting
:TRAC[*c*]:FREE?	Returns the available size (*available*) and the total size (*total*) of the trace buffer. Response is *available*,*total*.	
:TRAC[*c*]:POIN *points* :TRAC[*c*]:POIN? [MINimum|MAXimum|DEFault]	Sets the size of the trace buffer. This command is effective when the trace buffer control mode is set to NEV by the :TRAC[*c*]:FEED:CONT command. *points*=MINimum|MAXimum|DEFault|1 to 100000	100000
:TRAC[*c*]:POIN:ACT?	Returns the number of data in the trace buffer.	
:TRAC[*c*]:STAT:DATA?	Returns the result of the statistical operation for the data stored in the trace buffer.	
:TRAC[*c*]:STAT:FORM *operation* :TRAC[*c*]:STAT:FORM?	Selects the statistical operation performed by the :TRAC[*c*]:STAT:DATA? command. *operation*=MINimum|MAXimum|MEAN|SDEViation|PKPK	MEAN
:TRAC[*c*]:TST:FORM *rule* :TRAC[*c*]:TST:FORM?	Selects the rule for reading the timestamp data in the trace buffer. *rule*=DELTa|ABSolute	ABS

Using Advanced Functions

Table 2-17 HCOPy Subsystem

Command	Summary	Reset setting
:HCOP:SDUM:DATA?	Returns the data of the front panel screen image. The response is a definite length arbitrary binary block.	
:HCOP:SDUM:FORM *format* :HCOP:SDUM:FORM?	Sets the format of the image data. *format*=JPG|BMP|PNG|WMF	JPG

Table 2-18 DISPlay Subsystem

Command	Summary	Reset setting
:DISP:CSET *color* :DISP:CSET?	Selects the color set of the front panel display. *color*=1 (default color set)|2 (alternative color set)	
:DISP:DIG *digits* :DISP:DIG? [MINimum|MAXimum|DEFault]	Sets the display resolution of the data displayed on the front panel display. *digits*=4 (3½ digits)|5 (4½ digits)|6 (5½ digits)|7 (6½ digits)|MINimum|MAXimum|DEFault.	7
:DISP:ENAB *mode* :DISP:ENAB?	Enables or disables the front panel display under remote operation. *mode*=1|ON|0|OFF	
:DISP:VIEW *mode* :DISP:VIEW?	Sets the display mode, single 1, single 2, dual, graph, or roll. *mode*=SINGle1|SINGle2|DUAL|GRAPh|ROLL	SING1 for 1-ch models DUAL for 2-ch models
:DISP[:WIND[*d*]]:DATA?	Returns the data displayed on the front panel display.	

Command	Summary	Reset setting
:DISP[:WIND[*d*]]:TEXT:DATA *text* :DISP[:WIND[*d*]]:TEXT:DATA?	Sets the text message displayed on the center of the upper or lower display area of the front panel display. *text*: Up to 32 ASCII characters.	“”
:DISP[:WIND[*d*]]:TEXT:STAT :DISP[:WIND[*d*]]:TEXT:STAT?	Shows or hides the text message set by the :DISP[:WIND[*d*]]:TEXT:DATA command. *mode*=1|ON|0|OFF	OFF
:DISP:ZOOM *mode* :DISP:ZOOM?	Enables or disables the zoom function of the front panel display. *mode*=1|ON|0|OFF	OFF

Table 2-19 MMEMory Subsystem

Command	Summary	Reset setting
:MMEM:CAT? [*directory*]	Returns the memory usage and availability. Also returns the list of files and folders in the current specified directory. *directory*=<path>|USB:\<path>	
:MMEM:CDIR *directory* :MMEM:CDIR?	Changes the current directory to the specified directory. *directory*=<path>|USB:\<path>	USB:\
:MMEM:COPY *source,destination*	Makes a copy of an existing file in the current directory. *source*: Source file name. *destination*: Copy file name. Or directory name, <path>|USB:\<path>.	
:MMEM:DEL *file_name*	Deletes a file in the current directory. *file_name*: Name of the file to delete.	

Command	Summary	Reset setting
:MMEM:LOAD:MACR macro,*file_name*	Loads a macro from the specified file in the current directory. *macro*: Name of macro. *file_name*: Name of the file which contains the macro.	
:MMEM:LOAD:STAT *file_name*	Loads an instrument setup from the specified file in the current directory. *file_name*: Name of the file which contains the instrument setup.	
:MMEM:MDIR *directory*	Creates a new directory. *directory*=<path>|USB:\<path>	
:MMEM:MOVE *source*,*destination*	Moves or renames an existing file in the current directory. *source*: Source file name. *destination*: New file name. Or directory name, <path>|USB:\<path>.	
:MMEM:RDIR *directory*	Removes the specified empty directory. *directory*=<path>|USB:\<path>	
:MMEM:STOR:DATA[:ALL] *file*[,*chlist*] :MMEM:STOR:DATA:LIM *file*[,*chlist*] :MMEM:STOR:DATA:MATH *file*[,*chlist*] :MMEM:STOR:DATA:SENS *file*[,*chlist*]	Saves the limit test data, math expression result data, sense data, or all of these data for the specified channel to the specified file in the current directory. *file*: Name of the file to save the specified data. *chlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	(@1) for 1-ch models (@1,2) for 2-ch models

Command	Summary	Reset setting
:MMEM:STOR:MACR macro,*file_name*	Saves the macro to the specified file in the current directory. *macro*: Name of macro. *file_name*: Name of the file to save the macro.	
:MMEM:STOR:STAT *file_name*	Saves the instrument setup to the specified file in the current directory. *file_name*: Name of the file to save the instrument setup.	
:MMEM:STOR:TRAC *file_name*[,*chlist*]	Saves all data in the trace buffer for the specified channel to the specified file in the current directory. *file_name*: Name of the file to save the specified data. *chlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	(@1) for 1-ch models (@1,2) for 2-ch models

Table 2-20 PROGram Subsystem

Command	Summary	Reset setting
:PROG:CAT?	Returns the names of all programs defined in the program memory.	
:PROG:PON:COPY *name*	Specifies the power-on program. *name*: Name of the program used for the power-on program.	
:PROG:PON:DEL	Clears the power-on program.	
:PROG:PON:RUN *mode*	Enables or disables the power-on program. *mode*=1|ON|0|OFF	OFF

Command	Summary	Reset setting
:PROG[:SEL]:APP *program_code*	Adds a program code to the end of a program stored in the program memory. *program_code*: Program code. Up to 256 byte per execution. Sum of all program size in the program memory must be up to 100 KB.	
:PROG[:SEL]:DEF *program_code* :PROG[:SEL]:DEF?	Defines a program in the program memory by entering the initial program code. *program_code*: Program code. Up to 256 byte per execution. Sum of all program size in the program memory must be up to 100 KB. Maximum of 100 programs can be memorized.	
:PROG[:SEL]:DEL:ALL	Deletes all programs stored in the program memory.	
:PROG[:SEL]:DEL[:SEL]	Deletes a program stored in the program memory.	
:PROG[:SEL]:EXEC	Executes a program stored in the program memory.	
:PROG[:SEL]:NAME *name* :PROG[:SEL]:NAME?	Selects the program for performing the action by the following commands. If *name* does not specify the program stored in the program memory, this command creates a new program with the specified name and selects the program. If *name* specifies an existing program, this command selects the program.	

Command	Summary	Reset setting
:PROG[:SEL]:STAT *operation* :PROG[:SEL]:STAT?	Changes the execution status of a program stored in the program memory. *operation*=RUN|PAUSe|STEP|STOP|CONTinue	
:PROG[:SEL]:WAIT? *timeout*	Blocks other commands until the program execution status changes to Paused or Stopped. *timeout*: Timeout value, in seconds.	
:PROG:VAR[*h*] *value* :PROG:VAR[*h*]?	Sets a value to the variable specified by *h*. The variable is used in the program memory. A variable can be used in a program as %*h*% (*h*: integer. 1 to 100). *value*: Value of the variable specified by *h*. Up to 32 ASCII characters.	

Table 2-21 SYSTem Subsystem

Command	Summary	Reset setting
:SYST:BEEP[:IMM] *frequency*,*time*	Generates a beep sound of the specified frequency and duration. *frequency*=55 to 6640 Hz *time*=0.05 to 12.75 seconds	
:SYST:BEEP:STAT *mode* :SYST:BEEP:STAT?	Enables or disables the beeper. *mode*=0|OFF|1|ON	
:SYST:COMM:ENAB *mode*,*interface* :SYST:COMM:ENAB? *interface*	Enables or disables the remote interface GPIB, USB, or LAN, the remote service Sockets, Telnet, VXI-11, HiSLIP, or the built-in Web Interface. The setting is effective after rebooting the instrument. *mode*=0|OFF|1|ON *interface*=GPIB|USB|LAN|SOCKets|TELNet|VXI11|HISLip|WEB	

Command	Summary	Reset setting
:SYST:COMM:GPIB[:SELF]:ADDR *address* :SYST:COMM:GPIB[:SELF]:ADDR?	Sets the GPIB address of the instrument. *address*=0 to 30	
:SYST:COMM:LAN:ADDR *address* :SYST:COMM:LAN:ADDR? [CURR|STAT]	Sets the static LAN (IP) address of the instrument. The setting is enabled by the :SYST:COMM:LAN:UPD command. *address=A.B.C.D*, 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 225. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:COMM:LAN:BST?	Returns the LAN boot status of the instrument. Response is LAN_AUTO_IP, LAN_DHCP, LAN_FAULT, or LAN_STATIC.	
:SYST:COMM:LAN:CONT? :SYST:COMM:TCP:CONT?	Returns the control connection port number of the specified port.	
:SYST:COMM:LAN:DHCP *mode* :SYST:COMM:LAN:DHCP?	Enables or disables the use of the Dynamic Host Configuration Protocol (DHCP). The setting is enabled by the :SYST:COMM:LAN:UPD command. *mode*=0|OFF|1|ON	
:SYST:COMM:LAN:DNS[*c*] *address* :SYST:COMM:LAN:DNS[*c*]? [CURR|STAT]	Sets the IP address of the DNS server. *address=A.B.C.D*, 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:COMM:LAN:DOM?	Returns the domain name of the network to which the instrument is connected.	

Command	Summary	Reset setting
:SYST:COMM:LAN:GAT *address* :SYST:COMM:LAN:GATE *address* :SYST:COMM:LAN:GAT? [CURR|STAT] :SYST:COMM:LAN:GATE? [CURR|STAT]	Sets the IP address of the default gateway. The setting is enabled by the :SYST:COMM:LAN:UPD command. *address*=*A.B.C.D*, 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 225. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:COMM:LAN:HNAM *hostname* :SYST:COMM:LAN:HOST *hostname* :SYST:COMM:LAN:HNAM? [CURR|STAT] :SYST:COMM:LAN:HOST? [CURR|STAT]	Sets the host name of the instrument. The setting is enabled by the :SYST:COMM:LAN:UPD command. *hostname*: Host name. Up to 15 characters. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:COMM:LAN:MAC?	Returns the MAC address of the instrument.	
:SYST:COMM:LAN:SMAS *subnet_mask* :SYST:COMM:LAN:SMAS? [CURR|STAT]	Sets the static subnet mask. The setting is enabled by the :SYST:COMM:LAN:UPD command. *subnet_mask*=*A.B.C.D*, 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:COMM:LAN:TELN:PROM *prompt* :SYST:COMM:LAN:TELN:PROM?	Sets the command prompt displayed during a Telnet session for establishing communication with the instrument. *prompt*: Command prompt. Up to 15 characters.	

Command	Summary	Reset setting
:SYST:COMM:LAN:TELN:WMES *message* :SYST:COMM:LAN:TELN:WMES?	Sets the welcome message displayed during a Telnet session when starting the communication with the instrument. *message*: Welcome message. Up to 63 characters.	
:SYST:COMM:LAN:UPD	Disconnects all active LAN and Web Interface connections, updates the LAN setup, and restarts the LAN interface with the new setup.	
:SYST:COMM:LAN:WINS[*c*] *address* :SYST:COMM:LAN:WINS[*c*]? [CURR|STAT]	Sets the IP address of the WINS server. *address=A.B.C.D*, 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. CURR: Present setup value STAT: Reserved value for the next startup	
:SYST:DATA:QUAN? [*chanlist*]	Returns the number of data for the specified channel in the data buffer. *chanlist*=(@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1)	
:SYST:DATE *year,month,day* :SYST:DATE?	Sets the date of the internal clock. *year*: Year. 4-digit integer. *month*: Month. Integer from 1 to 12. *day*: Day. Integer from 1 to 31.	
:SYST:ERR:ALL?	Reads and returns all items in the error/event queue, and clears the queue.	
:SYST:ERR:CODE:ALL?	Reads all items in the error/event queue, returns all codes, and clears the queue.	
:SYST:ERR:CODE[:NEXT]?	Reads and removes the top item in the error/event queue, and returns the top code.	

Command	Summary	Reset setting
:SYST:ERR:COUN?	Returns the number of items in the error/event queue.	
:SYST:ERR[:NEXT]?	Reads and removes the top item in the error/event queue, and returns the top code and message.	
:SYST:FAN:MODE *mode* :SYST:FAN:MODE?	Sets the fan control mode. *mode*=NORMal|RACK	
:SYST:GRO[:DEF] *chanlist* :SYST:GRO[:DEF]?	Defines the channel group. *grouplist*=(@1,2) for making group, or *grouplist*=(@1),(@2) for breaking group	
:SYST:GRO:RES	Releases the channel group.	
:SYST:INT:TRIP?	Returns if the interlock circuit is close or open. Response is 0 or 1 that indicates the interlock circuit is close or open, respectively	
:SYST:LANG *mode* :SYST:LANG?	Selects the B2900 control command set. If the setting is changed, the instrument will be automatically rebooted. *mode*="DEFault"|"2400"	
:SYST:LFR *frequency* :SYST:LFR?	Selects the line frequency. *frequency*=50 (for 50 Hz)|60 (for 60 Hz)	
:SYST:LOCK:NAME?	Returns the current I/O interface (the I/O interface in use by the querying computer).	
:SYST:LOCK:OWN?	Returns the I/O interface that currently has a lock.	
:SYST:LOCK:REL	Decrements the lock count by one, and may release the I/O interface from which the command is executed.	

Command	Summary	Reset setting
:SYST:LOCK:REQ?	Requests a lock of the current I/O interface.	
:SYST:PON *memory*	Specifies the power-on state. *memory*=RST|RCL0|RCL1|RCL2|RCL3|RCL4	RST
:SYST:PRES	Presets the instrument settings and the front panel display.	
:SYST:SET *data* :SYST:SET?	Sends or loads the instrument setup data. *data*: Instrument setup data. Parameter data type is a definite length arbitrary binary block.	
:SYST:TIME *hour*,*minute*,*second* :SYST:TIME?	Sets the time of the internal clock. *hour*: Hour. Integer from 0 to 23. *minute*: Minute. Integer from 0 to 59. *second*: Second. Integer from 0 to 59.	
:SYST:TIME:TIM:COUN?	Returns the present count of the timer.	
:SYST:TIME:TIM:COUN:RES:AUTO *mode* :SYST:TIME:TIM:COUN:RES:AUTO?	Enables or disables the automatic reset function of the timer. If this function is enabled, the timer count is reset when the initiate action occurs. *mode*=0|OFF|1|ON	ON
:SYST:TIME:TIM:COUN:RES[:IMM]	Resets the timer count immediately.	
:SYST:VERS?	Returns the version of the SCPI standard.	

Table 2-22 STATus Subsystem

Command	Summary	Reset setting
:STAT:MEAS:COND? :STAT:OPER:COND? :STAT:QUES:COND?	Returns the value of the measurement, operation, or questionable status condition register.	
:STAT:MEAS:ENAB *mask* :STAT:OPER:ENAB *mask* :STAT:QUES:ENAB *mask* :STAT:MEAS:ENAB? :STAT:OPER:ENAB? :STAT:QUES:ENAB?	Sets the measurement, operation, or questionable status enable register. mask=0 to 65535 (decimal)	0
:STAT:MEAS[:EVEN]? :STAT:OPER[:EVEN]? :STAT:QUES[:EVEN]?	Returns the value of the measurement, operation, or questionable status event register.	
:STAT:MEAS:NTR *filter* :STAT:OPER:NTR *filter* :STAT:QUES:NTR *filter* :STAT:MEAS:NTR? :STAT:OPER:NTR? :STAT:QUES:NTR?	Sets the negative transition filter in the measurement, operation, or questionable status register. If you set a bit of the filter, a 1-to-0 transition of its register bit sets the corresponding bit of the event register. filter=0 to 65535 (decimal)	0
:STAT:MEAS:PTR *filter* :STAT:OPER:PTR *filter* :STAT:QUES:PTR *filter* :STAT:MEAS:PTR? :STAT:OPER:PTR? :STAT:QUES:PTR?	Sets the positive transition filter in the measurement, operation, or questionable status register. If you set a bit of the filter, a 0-to-1 transition of its register bit sets the corresponding bit of the event register. filter=0 to 65535 (decimal)	32767
:STAT:PRES	Sets all defined bits in the status system's PTR registers and clears the all bits in the NTR and Enable registers. The registers are returned to the default condition.	

Command	Summary	Reset setting
:STAT:QUES:CAL:COND? :STAT:QUES:CURR:COND? :STAT:QUES:TEMP:COND? :STAT:QUES:TEST:COND? :STAT:QUES:VOLT:COND?	Returns the value of the questionable status condition register.	
:STAT:QUES:CAL:ENAB *mask* :STAT:QUES:CURR:ENAB *mask* :STAT:QUES:TEMP:ENAB *mask* :STAT:QUES:TEST:ENAB *mask* :STAT:QUES:VOLT:ENAB *mask* :STAT:QUES:CAL:ENAB? :STAT:QUES:CURR:ENAB? :STAT:QUES:TEMP:ENAB? :STAT:QUES:TEST:ENAB? :STAT:QUES:VOLT:ENAB?	Sets the questionable calibration, current, temperature, test, or voltage status enable register. mask=0 to 65535 (decimal)	0

Command	Summary	Reset setting
:STAT:QUES:CAL[:EVEN]? :STAT:QUES:CURR[:EVEN]? :STAT:QUES:TEMP[:EVEN]? :STAT:QUES:TEST[:EVEN]? :STAT:QUES:VOLT[:EVEN]?	Returns the value of the questionable calibration, current, temperature, test, or voltage status event register.	
:STAT:QUES:CAL:NTR *filter* :STAT:QUES:CURR:NTR *filter* :STAT:QUES:TEMP:NTR *filter* :STAT:QUES:TEST:NTR *filter* :STAT:QUES:VOLT:NTR *filter* :STAT:QUES:CAL:NTR? :STAT:QUES:CURR:NTR? :STAT:QUES:TEMP:NTR? :STAT:QUES:TEST:NTR? :STAT:QUES:VOLT:NTR?	Sets the negative transition filter in the questionable calibration, current, temperature, test, or voltage status register. If you set a bit of the filter, a 1-to-0 transition of its register bit sets the corresponding bit of the event register. filter=0 to 65535 (decimal)	0
:STAT:QUES:CAL:PTR *filter* :STAT:QUES:CURR:PTR *filter* :STAT:QUES:TEMP:PTR *filter* :STAT:QUES:TEST:PTR *filter* :STAT:QUES:VOLT:PTR *filter* :STAT:QUES:CAL:PTR? :STAT:QUES:CURR:PTR? :STAT:QUES:TEMP:PTR? :STAT:QUES:TEST:PTR? :STAT:QUES:VOLT:PTR?	Sets the positive transition filter in the questionable calibration, current, temperature, test, or voltage status register. If you set a bit of the filter, a 0-to-1 transition of its register bit sets the corresponding bit of the event register. filter=0 to 65535 (decimal)	32767

3 Common Commands

This chapter describes common commands and queries of *IEEE 488.2*. The commands available for Agilent B2900 are listed in Table 3-1.

Table 3-1 Common Commands Available for B2900

Mnemonic	Name
*CAL?	Calibration query
*CLS	Clear status
*ESE	Standard event status enable command (query)
*ESR?	Standard event status register query
*IDN?	Identification query
*OPC	Operation complete command (query)
*RCL	Recall command
*RST	Reset command
*SAV	Save command
*SRE	Service request enable command (query)
*STB?	Read status byte query
*TRG	Trigger command
*TST?	Self-test query
*WAI	Wait-to-continue command

*CAL?

This query command performs the self-calibration, and returns the execution result.

Execution Conditions

Open the measurement terminals before starting the self-calibration.

Syntax

*CAL?

Query response

result <newline><^END>

result is 0 or 1 that indicates the calibration result. Response data type is NR1.

0: Passed

1: Failed

*CLS

This command clears the Status Byte register, the Standard Event Status register, and the Error Queue. This command does not clear the enable registers. For the SCPI status system, see “Status System Diagram” on page 1-19.

Also, this command stops the monitoring of pending operations by the *OPC command.

This command does not have query form.

Syntax

*CLS

*ESE

This command sets the bits of the Standard Event Status Enable register. This command programs the Standard Event Status Enable register bits. The programming determines which events of the Standard Event Status Enable register are allowed to set the ESB (Event Summary Bit) of the Status Byte register. A 1 in the bit position enables the corresponding event. For the SCPI status system, see “Status System Diagram” on page 1-19.

Syntax

*ESE *enable_number*

*ESE?

Parameter

enable_number Decimal value that is the sum of the binary-weighted values for the desired bits, hexadecimal, octal, or binary value. Parameter data type is NR1 or NDN.

Query response

enable_number <newline><^END>

enable_number is the sum of the binary-weighted values of the Enable register bits. The return format can be selected by the :FORMat:SREGister command. Response data type is NR1 or NDN.

Remarks

Bit definitions of the Standard Event register are shown in Table 3-2. All of the enabled events of the Standard Event Status Enable register are logically ORed to cause the Event Summary Bit (ESB) of the Status Byte register to be set.

The *CLS (clear status) command will not clear the enable register but it does clear all bits in the event register.

The :STATus:PRESet command does not clear the bits in the Status Byte register.

See Also

:FORMat:SREGister and :STATus:PRESet

*ESR?

This query returns the present contents of the Standard Event Status register. The event register is a read-only register, which stores (latches) all standard events. Reading the Standard Event Status Enable register clears it. For the SCPI status system, see “Status System Diagram” on page 1-19.

Syntax

*ESR?

Query response

register <newline><^END>

register is the binary-weighted sum of all bits set in the register. For example, if bit 3 (decimal value = 8) and bit 7 (decimal value = 128) are enabled, the query command will return 136. Response data type is NR1.

Remarks

Bit definitions of the Standard Event register are shown in Table 3-2.

To be reported to the Standard Event register, the corresponding bits in the event register must be enabled using the *ESE command.

Once a bit is set, it remains set until cleared by reading the event register or the *CLS (clear status) command.

See Also

*ESE

Table 3-2 Standard Event Register Bit Definitions

bit	decimal value	description	definition
0	1	OPC (operation complete)	All commands prior to and including *OPC have been executed.
1	2	Not used	0 is returned.
2	4	QYE (query error)	The instrument tried to read the output buffer but it was empty. Or, a new command line was received before a previous query has been read. Or, both the input and output buffers are full.
3	8	DDE (device-dependent error)	A self-test or calibration error occurred (an error in the -300 range or any positive error has been generated). For a complete listing of the error messages, see Chapter 5, “Error Messages.”
4	16	EXE (execution error)	An execution error occurred (an error in the -200 range has been generated).
5	32	CME (command error)	A command syntax error occurred (an error in the -100 range has been generated).
6	64	Not used	0 is returned.
7	128	PON (power on)	Power has been turned off and on since the last time the event register was read or cleared.

*IDN?

This query command returns the instrument's (mainframe) identification string which contains four comma-separated fields.

Syntax

*IDN?

Query response

Agilent Technologies,*model*,*serial*,*revision* <newline><^END>

model: mainframe model number

serial: mainframe serial number

revision: firmware revision number

Response data type is AARD.

*OPC

This command starts to monitor pending operations, and sets/clears the operation complete (OPC) bit in the Standard Event Status register as follows.

- If there is no pending operation, the OPC bit is set to 1.
- If there are any pending operations, the OPC bit is set to 0. The bit will be set to 1 again when all pending operations are completed.

The *OPC command is required to enable the OPC bit. To stop monitoring pending operations (disable OPC bit), execute the *CLS command.

Other commands cannot be executed until this command completes.

Syntax

*OPC

*OPC?

Query response

1 <newline><^END>

The query returns 1 if the instrument has completed all pending operations sent before this command. Response data type is NR1.

See Also

*WAI

*RCL

This command restores the instrument to a state that was previously stored in one of the memory locations 0 through 9 with the *SAV command.

Syntax

*RCL *memory*

Parameter

memory One of the memory locations 0 to 9. Parameter data type is NR1.

Remarks

The device state stored in the location 0 is automatically recalled at power turn-on when the Output Power-On state is set to *RCL 0.

You cannot recall the instrument state from a storage location that is empty or was deleted. You can only recall a state from a location that contains a previously stored state.

The *RST command does not affect the configurations stored in memory. Once a state is stored, it remains until it is overwritten or specifically deleted.

*RST

This command performs an instrument reset. This command resets the volatile memory of the instrument to the initial setting.

Syntax

*RST

Remarks

This command cancels any measurement or output trigger actions presently in process, and resets the Waiting for arm and trigger bits in the Status Operation Condition register.

*SAV

This command stores the present state of the instrument to the specified location in non-volatile memory. Up to 10 states can be stored in the memory locations 0 through 9. Any state previously stored in the same location will be overwritten. Use the *RCL command to retrieve instrument states.

Syntax

*SAV *memory*

Parameter

memory	One of the memory locations 0 to 9. Parameter data type is NR1. The locations 0 to 4 are in the non-volatile memory, and the locations 5 to 9 are in the volatile memory.

Remarks

If a particular state is desired at power-on, it should be stored in the location 0. It will then be automatically recalled at power turn-on if the Output Power-On state is set to *RCL 0.

Data described in “Non-Volatile Settings” on page 1-23 is not affected by the *SAV command.

The *RST command does not affect the configurations stored in memory. Once a state is stored, it remains until it is overwritten or specifically deleted.

CAUTION

This command causes a write cycle to non-volatile memory. Non-volatile memory has a finite maximum number of write cycles. Programs that repeatedly cause write cycles to non-volatile memory can eventually exceed the maximum number of write cycles and cause the memory to fail.

*SRE

This command sets the value of the Service Request Enable register. This register determines which bits from the Status Byte register are summed to set the Master Status Summary (MSS) bit and the Request for Service (RQS) summary bit. A 1 in the bit position enables the corresponding event. For the SCPI status system, see “Status System Diagram” on page 1-19.

The query reads the enable register and returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

Syntax

*SRE *value*

*SRE?

Parameter

value Decimal value which corresponds to the binary-weighted sum of the bits in the register (see Table 3-3). Parameter data type is NRf.

For example, to enable bit 0 (decimal value = 1), bit 3 (decimal value = 8), and bit 6 (decimal value = 64), the corresponding decimal value would be 73 (1 + 8 + 64).

Query response

value <newline><^END>

value is the binary-weighted sum of all bits set in the register. For example, if bit 3 (decimal value = 8) and bit 7 (decimal value = 128) are enabled, the query command will return 136. Response data type is NR1.

Remarks

Bit definitions of the Status Byte register are shown in Table 3-3.

All of the enabled events of the Standard Event Status Enable register are logically ORed to cause the Event Summary Bit (ESB) of the Status Byte register to be set. All such enabled bits are then logically ORed to cause the MSS bit (bit 6) of the Status Byte register to be set.

When the controller conducts a serial poll in response to SRQ, the RQS bit is cleared, but the MSS bit is not. When *SRE is cleared (by programming it with 0), the power system cannot generate an SRQ to the controller.

The *CLS (clear status) command will not clear the enable register but it does clear all bits in the event register.

A :STATus:PRESet command does not clear the bits in the Status Byte register.

Table 3-3 Status Byte Register Bit Definitions

bit	decimal value	description	definition
0	1	Measurement status summary	One or more bits are set in the Measurement Status register (bits must be enabled, see :STATus:<MEASurement|OPERation|QUEStionable>:ENABle command).
1	2	Not used	0 is returned.
2	4	Error queue not empty	One or more errors have been stored in the Error Queue (see :SYSTem:ERRor[:NEXT]? command).
3	8	Questionable status summary	One or more bits are set in the Questionable Status register (bits must be enabled, see :STATus:<MEASurement|OPERation|QUEStionable>:ENABle command).
4	16	Output buffer	Data is available in the instrument's output buffer.
5	32	Event status byte summary	One or more bits are set in the Standard Event register (bits must be enabled, see *ESE command).
6	64	Master status summary (Request for service)	One or more bits are set in the Status Byte register (bits must be enabled, see *SRE command). Also used to indicate a request for service.
7	128	Operation status summary	One or more bits are set in the Operation Status register (bits must be enabled, see :STATus:<MEASurement|OPERation|QUEStionable>:ENABle command).

*STB?

This query reads the Status Byte register, which contains the status summary bits and the Output Queue MAV bit. The Status Byte register is a read-only register and the bits are not cleared when it is read. For the SCPI status system, see “Status System Diagram” on page 1-19.

Syntax

*STB?

Query response

register <newline><^END>

register is the binary-weighted sum of all bits set in the register. For example, if bit 1 (decimal value = 2) and bit 4 (decimal value = 16) are set (and the corresponding bits are enabled), this command will return 18. Response data type is NR1.

Remarks

Bit definitions of the Status Byte register are shown in Table 3-3.

The input summary bits are cleared when the appropriate event registers are read. The MAV bit is cleared at power-on, by *CLS, or when there is no more response data available.

A serial poll also returns the value of the Status Byte register, except that bit 6 returns Request for Service (RQS) instead of Master Status Summary (MSS). A serial poll clears RQS, but not MSS. When MSS is set, it indicates that the instrument has one or more reasons for requesting service.

*TRG

This common command generates a trigger when the trigger subsystem has BUS selected as its source. The command has the same affect as the Group Execute Trigger (GET) command.

Syntax

*TRG

*TST?

This query causes the instrument to do a self-test and report any errors. A 0 indicates the instrument passed self-test. If all tests pass, you can have a high confidence that the instrument is operational.

Syntax

*TST?

Query response

result <newline><^END>

result is 0 or 1 that indicates the self-test result. Response data type is NR1.

0: all tests passed

1: one or more tests failed

Remarks

If one or more tests fail, a 1 is returned and an error is stored in the error queue. For a complete listing of the error messages related to self-test failures, see Chapter 5, “Error Messages.”

If one or more tests fail, see the Service Guide for instructions on returning the instrument to Agilent for service.

*TST? also forces an *RST command.

*WAI

This command instructs the instrument not to process any further commands until all pending operations are completed. Pending operations are as defined under the *OPC command.

Syntax

*WAI

Remarks

*WAI can be aborted only by sending the instrument a Device Clear command.

See Also

*OPC

4 Subsystem Commands

This chapter describes subsystem commands available for Agilent B2900 in alphabetical order. There are the following subsystems.

- “CALCulate Subsystem”
- “DISPlay Subsystem”
- “FETCh Subsystem”
- “FORMat Subsystem”
- “HCOPy Subsystem”
- “LXI Subsystem”
- “MEASure Subsystem”
- “MMEMory Subsystem”
- “OUTPut Subsystem”
- “PROGram Subsystem”
- “READ Subsystem”
- “SENSe Subsystem”
- “SOURce Subsystem”
- “STATus Subsystem”
- “SYSTem Subsystem”
- “TRACe Subsystem”
- “TRIGger Subsystem”

CALCulate Subsystem

For the numeric suffixes [*c*], [*m*], and [*n*], see “Numeric Suffix” on page 1-8.

:CALCulate:CLIMits:CLEar:AUTO

Enables or disables the automatic clear function of the composite limit test.

Syntax

:CALCulate[*c*]:CLIMits:CLEar:AUTO *mode*

:CALCulate[*c*]:CLIMits:CLEar:AUTO?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the automatic clear function which clears the composite limit test results and ports (GPIO lines) automatically with each :INITiate command. See “:INITiate[:IMMediate]<:ACQuire|:TRANsient|[:ALL]>” on page 4-157.

mode=0 or OFF disables the automatic clear function. The composite limit test results and ports (GPIO lines) must be cleared manually before the next composite limit test is started. Execute the :CALCulate:CLIMits:CLEar[:IMMediate] command to clear them immediately.

Query response

mode <newline>

mode is 0 or 1, and indicates that the automatic clear function is off or on, respectively. Response data type is NR1.

Example

:CALC:CLIM:CLE:AUTO 1

:CALC2:CLIM:CLE:AUTO?

:CALCulate:CLIMits:CLEar:AUTO:DELay

Sets the delay time for the automatic clear of the composite limit test. See “:CALCulate:CLIMits:CLEar:AUTO” on page 4-3. The delay time is defined as the time before the automatic clear is performed after the measurement is completed.

Syntax

:CALCulate[*c*]:CLIMits:CLEar:AUTO:DELay *time*

:CALCulate[*c*]:CLIMits:CLEar:AUTO:DELay? [*time*]

Parameter ***time*** *value* (+1E-5 to 60 seconds)|MINimum|MAXimum|DEFault (default is +1E-4). Parameter data type is NRf+. Query does not support *time*=*value*. If you specify the value less than MIN or greater than MAX, *time* is automatically set to MIN or MAX.

Query response *time* <newline>

time returns the present setting of delay time for the automatic clear. If a parameter is specified, *time* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example :CALC:CLIM:CLE:AUTO:DEL 1E-3

:CALC2:CLIM:CLE:AUTO:DEL?

:CALCulate:CLIMits:CLEar[:IMMediate]

Clears the composite limit test results and ports (GPIO lines) immediately.

Syntax :CALCulate[*c*]:CLIMits:CLEar[:IMMediate]

Example :CALC:CLIM:CLE:IMM

:CALC2:CLIM:CLE

:CALCulate:CLIMits:<FAIL|PASS>:DIGital[:DATA]

Defines a fail/pass pattern that appears near the end of the flowcharts shown in Figures 2-10 and 2-11. This is a bit pattern used to indicate the composite limit test result (fail or pass). It must be entered in the format set by the :FORMat:DIGital command.

Syntax :CALCulate[*c*]:CLIMits:<FAIL|PASS>:DIGital[:DATA] *bit_pattern*

:CALCulate[*c*]:CLIMits:<FAIL|PASS>:DIGital[:DATA]?

For <FAIL|PASS>, specify PASS for a pass pattern or FAIL for a fail pattern.

Parameter ***bit_pattern*** 0 (default setting in decimal expression) to 16383. Parameter data type is NR1 or NDN.

Query response *bit_pattern* <newline>

bit_pattern returns the fail/pass bit pattern in the format specified by the :FORMat:DIGital command. Response data type is NR1 or NDN.

Example

:CALC:CLIM:FAIL:DIG:DATA 64

:CALC2:CLIM:PASS:DIG?

:CALCulate:CLIMits:MODE

Sets the operation mode of the composite limit test to GRADing or SORTing.

Syntax

:CALCulate[*c*]:CLIMits:MODE *mode*

:CALCulate[*c*]:CLIMits:MODE?

Parameter

mode SORT (sorting)|GRAD (grading, default). Parameter data type is CPD.

mode=GRAD performs limit tests for up to 12 test limits until a failure is detected. See Figure 2-11 for an example of a flowchart under the grading mode.

mode=SORT performs limit tests for up to 12 test limits until a pass is detected. See Figure 2-10 for an example of a flowchart under the sorting mode.

A limit test is a pass/fail judgement performed during a composite limit test. It is performed at the “Pass?” step in Figures 2-10 and 2-11.

Query response

mode <newline>

mode returns GRAD or SORT. Response data type is CPD.

Example

:CALC:CLIM:MODE SORT

:CALC2:CLIM:MODE?

:CALCulate:CLIMits:STATe

Enables or disables the composite limit test.

Syntax

:CALCulate[*c*]:CLIMits:STATe *mode*

:CALCulate[*c*]:CLIMits:STATe?

Parameter

mode 1|ON (default)|0|OFF. Parameter data type is boolean.

mode=1 or ON enables the composite limit test.

mode=0 or OFF disables the composite limit test.

Query response

mode <newline>

mode is 0 or 1, and indicates that the composite limit test is off or on, respectively. Response data type is NR1 or NDN.

Example :CALC:CLIM:STAT 1

:CALC2:CLIM:STAT?

:CALCulate:CLIMits:STATe:ANY?

Checks if the present composite limit test contains a limit test, which is a pass/fail judgement. It is performed at the “Pass?” step in Figures 2-10 and 2-11.

Syntax :CALCulate[*c*]:CLIMits:STATe:ANY?

Query response *status* <newline>

status returns 0 or 1. Response data type is NR1.

0: No limit test exists.

1: At least one limit test exists.

Example :CALC2:CLIM:STAT:ANY?

:CALCulate:CLIMits:UPDate

Only for the GRAD composite limit test. Enables or disables the immediate result output or update. See “Immediate?” shown in Figure 2-11.

When enabled, the result output/update is executed immediately when the first failure or all pass is detected. The result is the pass/fail bit pattern defined by the :CALCulate:CLIMits:<FAIL|PASS>:DIGital[:DATA] command. If all pass is detected, the pattern will be the pass pattern.

Syntax :CALCulate[*c*]:CLIMits:UPDate *result*

:CALCulate[*c*]:CLIMits:UPDate?

Parameter ***result*** END|IMMediate (default). Parameter data type is CPD.

result=IMM enables immediate result output.

result=END disables immediate result output.

Query response *result* <newline>

result returns IMM or END. Response data type is CRD.

Example

:CALC:CLIM:UPD END

:CALC2:CLIM:UPD?

:CALCulate:DATA?

Returns limit test data. Elements of the returned data are specified by the :FORMat:ELEMents:CALCulate command. The limit test data can be expressed by the following formula.

limit test data = input data - null offset

input data: Data specified by :CALCulate:FEED

null offset: Data set by :CALCulate:OFFSet or :CALCulate:OFFSet:ACQuire

If the null offset function is disabled by the :CALCulate:OFFSet:STATe command, *null offset*=0.

Syntax

:CALCulate[*c*]:DATA? [*offset*[, *size*]]

Parameter

offset Indicates the beginning of the data received. *n*|CURRent|STARt (default). Parameter data type is NR1 or CPD.

offset=*n* specifies the *n+1*th data. *n* is an integer, 0 to maximum (depends on the buffer state).

offset=CURR specifies the present data position.

offset=STAR specifies the top of the data buffer. Same as *offset*=0.

size Number of data to be received. 1 to maximum (depends on the buffer state). Parameter data type is NR1. If this parameter is not specified, all data from *offset* is returned.

Query response

data <newline>

Response data type is NR3. See “Data Output Format” on page 1-12.

Example

:CALC2:DATA? 0,10

:CALCulate:DATA:LATest?

Returns the latest limit test data. Elements of the returned data are specified by the :FORMat:ELEMents:CALCulate command. The limit test data can be expressed by the following formula.

limit test data = *input data* - *null offset*

input data: Data specified by :CALCulate:FEED

null offset: Data set by :CALCulate:OFFSet or :CALCulate:OFFSet:ACQuire

If the null offset function is disabled by the :CALCulate:OFFSet:STATe command, *null offset*=0.

Syntax

:CALCulate[*c*]:DATA:LATest?

Query response

data <newline>

Response data type is NR3. See “Data Output Format” on page 1-12.

Example

:CALC2:DATA:LAT?

:CALCulate:DIGital:BIT

Assigns the GPIO pins used for the result output. The result is the pass/fail bit pattern defined by the :CALCulate:CLIMits:<FAIL|PASS>:DIGital[:DATA] command.

Syntax

:CALCulate[*c*]:DIGital:BIT *pin*

:CALCulate[*c*]:DIGital:BIT?

Parameter

pin EXT*n*|NONE (default). Parameter data type is CPD. EXT*n* specifies a GPIO pin, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

pin=NONE does not assign the GPIO pins.

To assign the GPIO pins, *pin* must be a comma separated EXT string like `EXT`*n*`,EXT`*n*+*1*`,` .... , and `EXT`*n* must be LSB. The specified pins must be continuous. For example, *pin*=`EXT9,EXT10,EXT11` is effective for this command, and EXT9, EXT10, and EXT11 are assigned to BIT0 (LSB), BIT1, and BIT2, respectively. On the contrary, non-continuous pin assignment such as *pin*=`EXT9,EXT10,EXT14` is not effective.

Query response *pin* <newline>

pin returns NONE or a comma separated EXT string. Response data type is CRD.

Example :CALC:DIG:BIT EXT9,EXT10,EXT11

:CALC2:DIG:BIT?

:CALCulate:DIGital:<BUSY|EOT|SOT>

Assigns the GPIO pin for the BUSY (busy), EOT (end of test), or SOT (start of test) signal line for the composite limit test.

Syntax :CALCulate[*c*]:DIGital:<BUSY|EOT|SOT> *pin*

:CALCulate[*c*]:DIGital:<BUSY|EOT|SOT>?

For <BUSY|EOT|SOT>, specify BUSY for assigning the busy line, EOT for assigning the end of test line, or SOT for assigning the start of test line.

Parameter

pin	EXT*n*|NONE (default). Parameter data type is CPD. EXT*n* specifies a GPIO pin, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

To assign the GPIO pin, *pin* must be a EXT string like EXT*n*. For example, *pin*=EXT14.

pin=NONE does not assign the GPIO pin.

Query response *pin* <newline>

pin returns NONE or a EXT string. Response data type is CRD.

Example :CALC:DIG:EOT EXT14

:CALC2:DIG:SOT?

:CALCulate:FEED

Specifies the *input data* value used for calculating the limit test data. The limit test is a pass/fail judgement performed during a composite limit test. The limit test is performed at the “Pass?” step in Figures 2-10 and 2-11. The limit test data is returned by the :CALCulate:DATA? or :CALCulate:DATA:LATest? command.

Syntax :CALCulate[*c*]:FEED *type*

:CALCulate[*c*]:FEED?

Parameter

type Data type. MATH|RESistance|CURRent|VOLTage (default). Parameter data type is CPD.

type=VOLT specifies the voltage measurement data.

type=CURR specifies the current measurement data.

type=RES specifies the resistance calculation data given by the following formula.

Resistance=Vmeas/Imeas

Where, Vmeas is the voltage measurement data, and Imeas is the current measurement data.

type=MATH specifies the data given by a math expression. The math expression must be specified before the :CALC:FEED MATH command is executed.

An existing math expression can be specified by the :CALCulate:MATH[:EXPRession]:NAME command.

A new math expression can be defined by the :CALCulate:MATH[:EXPRession]:NAME and :CALCulate:MATH[:EXPRession][:DEFine] commands.

Query response

type <newline>

type returns the present setting of data type, MATH, RES, CURR, or VOLT. Response data type is CRD.

Example

:CALC:FEED MATH

:CALC2:FEED?

:CALCulate:LIMit:COMPliance:DIGital[:DATA]

Defines the bit pattern used to indicate a *failure* of the compliance status check specified by *m*. It must be entered in the format set by the :FORMat:DIGital command.

Syntax

:CALCulate[*c*]:LIMit[*m*]:COMPliance:DIGital[:DATA] *bit_pattern*

:CALCulate[*c*]:LIMit[*m*]:COMPliance:DIGital[:DATA]?

Parameter

bit_pattern 0 (default setting in decimal expression) to 16383. Parameter data type is NR1 or NDN.

Query response *bit_pattern* <newline>

bit_pattern returns the fail bit pattern in the format specified by the :FORMat:DIGital command. Response data type is NR1 or NDN.

Example :CALC:LIM:COMP:DIG:DATA 64

:CALC2:LIM12:COMP:DIG?

:CALCulate:LIMit:COMPliance:FAIL

Sets the judgement criteria for the compliance status check specified by *m*.

Syntax :CALCulate[*c*]:LIMit[*m*]:COMPliance:FAIL *criteria*

:CALCulate[*c*]:LIMit[*m*]:COMPliance:FAIL?

Parameter ***criteria*** OUT|IN (default). Parameter data type is CPD.

criteria=IN determines that the limit test has failed if the channel goes into the compliance state.

criteria=OUT determines that the limit test has failed if the channel comes out of the compliance state.

Query response *criteria* <newline>

criteria returns IN or OUT which indicates the present setting of the judgement criteria. Response data type is CRD.

Example :CALC:LIM:COMP:FAIL OUT

:CALC2:LIM12:COMP:FAIL?

:CALCulate:LIMit:FAIL?

Returns the result of the limit test specified by *m*.

Syntax :CALCulate[*c*]:LIMit[*m*]:FAIL?

Query response *result* <newline>

result returns 0 or 1. Response data type is NR1.

0: Passed

1: Failed

Example :CALC2:LIM12:FAIL?

:CALCulate:LIMit:FUNCtion

Sets the type of the limit test specified by *m*.

Syntax :CALCulate[*c*]:LIMit[*m*]:FUNCtion *type*

:CALCulate[*c*]:LIMit[*m*]:FUNCtion?

Parameter ***type*** COMPliance|LIMit (default). Parameter data type is CPD.

type=COMP sets the compliance status check which checks if the channel is in the compliance status.

type=LIM sets the limit test which checks if the measurement value is between the upper limit and the lower limit.

Query response *type* <newline>

type returns the present setting of the type, COMP or LIM. Response data type is CRD.

Example :CALC:LIM:FUNC COMP

:CALC2:LIM12:FUNC?

:CALCulate:LIMit:<LOWer|UPPer>

Sets a lower/upper limit used for the limit test specified by *m*.

Syntax :CALCulate[*c*]:LIMit[*m*]:<LOWer|UPPer> *limit*

:CALCulate[*c*]:LIMit[*m*]:<LOWer|UPPer>? [*limit*]

For <LOWer|UPPer>, specify LOWer for lower limit, or UPPer for upper limit.

Parameter ***limit*** *value* (-9.999999E+20 to +9.999999E+20)|MINimum|MAXimum|DEFault (default is -1 for the lower limit and +1 for the upper limit). Parameter data type is NRf+. Query does not support *limit=value*.

Query response *limit* <newline>

limit returns the present setting of the lower/upper limit used for the limit test specified by *m*. If a parameter is specified, *limit* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:CALC:LIM:LOW -2.5

:CALC2:LIM12:UPP?

:CALCulate:LIMit:<LOWer|UPPer>:DIGital[:DATA]

Defines the bit pattern used to indicate *failed-by-exceeding-lower-limit* or *failed-by-exceeding-upper-limit* of the limit test specified by *m*. It must be entered in the format set by the :FORMat:DIGital command. The bit pattern defined by this command is used for the GRAD composite limit test.

Syntax

:CALCulate[*c*]:LIMit[*m*]:<LOWer|UPPer>:DIGital[:DATA] *bit_pattern*

:CALCulate[*c*]:LIMit[*m*]:<LOWer|UPPer>:DIGital[:DATA]?

For <LOWer|UPPer>, specify LOWer for *failed-by-exceeding-lower-limit*, or UPPer for *failed-by-exceeding-upper-limit*.

Parameter

bit_pattern 0 (default setting in decimal expression) to 16383. Parameter data type is NR1 or NDN.

Query response

bit_pattern <newline>

bit_pattern returns the fail bit pattern in the format specified by the :FORMat:DIGital command. Response data type is NR1 or NDN.

Example

:CALC:LIM:LOW:DIG:DATA 64

:CALC2:LIM12:UPP:DIG?

:CALCulate:LIMit:PASS:DIGital[:DATA]

Defines the bit pattern used to indicate a *pass* of the limit test specified by *m*. It must be entered in the format set by the :FORMat:DIGital command. The bit pattern defined by this command is used for the SORT composite limit test.

Syntax

:CALCulate[*c*]:LIMit[*m*]:PASS:DIGital[:DATA] *bit_pattern*

:CALCulate[*c*]:LIMit[*m*]:PASS:DIGital[:DATA]?

Parameter ***bit_pattern*** 0 (default setting in decimal expression) to 16383. Parameter data type is NR1 or NDN.

Query response *bit_pattern* <newline>

bit_pattern returns the pass bit pattern in the format specified by the :FORMat:DIGital command. Response data type is NR1 or NDN.

Example :CALC:LIM:PASS:DIG:DATA 64

:CALC2:LIM12:PASS:DIG?

:CALCulate:LIMit:STATe

Enables or disables the limit test specified by *m*.

Syntax :CALCulate[*c*]:LIMit[*m*]:STATe *mode*

:CALCulate[*c*]:LIMit[*m*]:STATe?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the limit test specified by *m*.

mode=0 or OFF disables the limit test specified by *m*.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the limit test is off or on, respectively. Response data type is NR1.

Example :CALC:LIM:STAT 1

:CALC2:LIM12:STAT?

:CALCulate:MATH:DATA?

Returns the calculation result data. Elements of the returned data are specified by the :FORMat:ELEMents:CALCulate command. Math expression for the calculation is defined by the :CALCulate:MATH[:EXPRession]:NAME and :CALCulate:MATH[:EXPRession][:DEFine]commands.

Syntax :CALCulate[*c*]:MATH:DATA? [*offset*[, *size*]]

Parameter

offset	Indicates the beginning of the data received. *n*|CURRent|STARt (default). Parameter data type is NR1 or CPD. *offset*=*n* specifies the *n+1*th data. *n* is an integer, 0 to maximum (depends on the buffer state). *offset*=CURR specifies the present data position. *offset*=STAR specifies the top of the data buffer. Same as *offset*=0.
size	Number of data to be received. 1 to maximum (depends on the buffer state). Parameter data type is NR1. If this parameter is not specified, all data from *offset* is returned.

Query response *data* <newline>

Response data type is NR3.

See “Data Output Format” on page 1-12.

Example :CALC2:MATH:DATA? 0,10

Remarks If the math expression contains some measurement results, measurement may be performed many times to obtain the result. For example, measurement must be performed twice to get the result of the following math expression.

math expression =(CURR[1]-CURR[0])

:CALCulate:MATH:DATA:LATest?

Returns the latest calculation result data. Elements of the returned data are specified by the :FORMat:ELEMents:CALCulate command. Math expression for the calculation is defined by the :CALCulate:MATH[:EXPRession]:NAME and :CALCulate:MATH[:EXPRession][:DEFine]commands.

Syntax :CALCulate[*c*]:MATH:DATA:LATest?

Query response *data* <newline>

Response data type is NR3. See “Data Output Format” on page 1-12.

Example :CALC2:MATH:DATA:LAT?

:CALCulate:MATH[:EXPRession]:CATalog?

Returns the list of all the predefined and user-defined math expression names.

Syntax

:CALCulate[*c*]:MATH[:EXPRession]:CATalog?

Query response

catalog <newline>

catalog returns all of the predefined and user-defined math expression names. Response data type is AARD. For example, if the instrument stores the math expressions POWER, OFFCOMPOHM, VOLTCOEF, and VARALPHA, *catalog* returns “POWER”,“OFFCOMPOHM”,“VOLTCOEF”,“VARALPHA”.

Example

:CALC2:MATH:EXPR:CAT?

:CALCulate:MATH[:EXPRession][:DEFine]

Defines a math expression which will be a user-defined math expression. For the resources effective for the expression, see “Resources used in the expressions” on page 4-17. Also see “Predefined math expressions” on page 4-18 for the definition of predefined expressions.

Before executing this command, the math expression must be selected by the:CALCulate:MATH[:EXPRession]:NAME command.

Syntax

:CALCulate[*c*]:MATH[:EXPRession][:DEFine] *definition*

:CALCulate[*c*]:MATH[:EXPRession][:DEFine]?

Parameter

definition Definition of a math expression. Up to 256 ASCII characters. Parameter data type is Expr. The expression must be enclosed by parentheses. For example, *definition*=(SOUR2/CURR2). Maximum of 32 math expressions can be defined including the predefined math expressions.

Query response

definition <newline>

definition returns the definition of the math expression currently selected. For example, *definition* returns (SOUR2/CURR2). Response data type is Expr.

Example

:CALC:MATH:EXPR:NAME “Expression_for_ch1”
:CALC:MATH:EXPR:DEF ((CURR[1]-CURR[0])*(RES[1]-RES[0]))

:CALC:MATH?

Resources used in the expressions

The following resources can be used in user-defined math expressions.

- Reserved variables

  The variables listed in Table 4-1 are reserved for reading the channel output or measurement data.

  Scalar variable is used for spot measurement data.

  Vector (array) variable is used for sweep measurement data.

- Math operators

  The following operators are available.

  - Arithmetic operators: +, -, *, /, ^, see Table 4-2
  - Elementary functions: ln, log, sin, cos, tan, exp

  The functions log and ln perform the operation after calculating the absolute value. So if a negative value is specified, they do not result in an error, but calculate as if a positive value was specified. For example, log(-10) results in log(10)=1.

- Numeric value

  Decimal (0 to 4294967294, 4294967295 indicates -1), binary (32 bit, 0 to 0b11111111111111111111111111111111), or hexadecimal (0 to 0xFFFFFFFF).

Table 4-1 Reserved Variables

Reserved variable [a]		Description
Scalar	**Vector**	
SOUR[*c*]	SOUR[*c*][]	Source output setting data
VOLT[*c*]	VOLT[*c*][]	Voltage measurement data
CURR[*c*]	CURR[*c*][]	Current measurement data
RES[*c*]	RES[*c*][]	Resistance measurement data
TIME[*c*]	TIME[*c*][]	Time (timestamp) data

a. The numeric suffix [*c*] is effective for specifying the channel. For example, use CURR2 to read the current spot measurement data for channel 2. See “Numeric Suffix” on page 1-8.

Table 4-2 **Arithmetic and Unary Operators**

Priority of task	Operator	Description
High : : : : : Low	( )	Parentheses
	+ and -	Unary plus operator and unary minus operator
	^	Exponentiation operator
	* and /	Multiplication operator and division operator
	+ and -	Additive operator and subtraction operator

Predefined math expressions

The following math expressions are already defined in the instrument. The predefined math expressions are not cleared by the power off/on operations.

- Power (POWER)
- Offset Compensated Ohms (OFFCOMPOHM)
- Varistor Alpha (VARALPHA)
- Voltage Coefficient (VOLTCOEF)

POWER

Calculates power using the following formula.

POWER = VOLT[*c*] * CURR[*c*]

OFFCOMPOHM

Calculates offset compensated ohms (resistance) using the following formula.

OFFCOMPOHM = (VOLT[*c*][1] - VOLT[*c*][0]) / (CURR[*c*][1] - CURR[*c*][0])

where, VOLT[*c*][0] and CURR[*c*][0] are data measured with the current output level, and VOLT[*c*][1] and CURR[*c*][1] are data measured with a different current output level or zero output.

This function is effective for reducing measurement errors in low resistance measurements.

VARALPHA

Calculates varistor alpha using the following formula.

VARALPHA = log(CURR[*c*][1] / CURR[*c*][0]) / log(VOLT[*c*][1] / VOLT[*c*][0])

where, CURR[*c*][0] and VOLT[*c*][0] are measurement data at a point on a varistor's non-linear I-V characteristics curve, and CURR[*c*][1] and VOLT[*c*][1] are data at another point.

VOLTCOEF Calculates voltage coefficient using the following formula.

VOLTCOEF = (RES[*c*][1] - RES[*c*][0]) / (RES[*c*][1] * (VOLT[*c*][1] - VOLT[*c*][0])) * 100 %

where, RES[*c*][0] and RES[*c*][1] are resistance measurement data at the first and second measurement points, respectively, and VOLT[*c*][0] and VOLT[*c*][1] are voltage measurement data at the first and second measurement points, respectively.

The voltage coefficient is known as the ratio of the fractional change for a resistor whose resistance varies with voltage.

:CALCulate:MATH[:EXPRession]:DELete:ALL

Deletes all user-defined math expressions. This command cannot delete predefined math expressions.

Syntax :CALCulate[*c*]:MATH[:EXPRession]:DELete:ALL

Example :CALC2:MATH:EXPR:DEL:ALL

:CALCulate:MATH[:EXPRession]:DELete[:SELected]

Deletes an user-defined math expression. This command cannot delete a predefined math expression.

Syntax :CALCulate[*c*]:MATH[:EXPRession]:DELete[:SELected] *name*

Parameter

name	Name of the math expression to delete. Up to 32 ASCII characters. Parameter data type is SPD.

Example :CALC2:MATH:EXPR:DEL:SEL “TempExpression1”

:CALCulate:MATH[:EXPRession]:NAME

Selects a math expression used for calculation. A predefined math expression or an user-defined math expression can be specified by the *name* parameter.

See “Predefined math expressions” on page 4-18 for the definition of predefined math expressions.

A new user-defined math expression can be added by executing this command with a new name, and executing the :CALCulate:MATH[:EXPRession][:DEFine] command with a new definition.

Existing user-defined math expression can be changed by executing this command with its name, and executing the :CALCulate:MATH[:EXPRession][:DEFine] command with a new definition.

Syntax

:CALCulate[*c*]:MATH[:EXPRession]:NAME *name*

:CALCulate[*c*]:MATH[:EXPRession]:NAME?

Parameter

name Name of a math expression. Up to 32 ASCII characters without any control characters, space characters, single and double quotes, and comma. Parameter data type is SPD.

Query response

name <newline>

name returns the name of the math expression currently selected. For example, *name* returns “Expression_for_ch2”. Response data type is SRD.

Example

:CALC2:MATH:EXPR:NAME “Expression_for_ch2”

:CALC2:MATH:NAME?

:CALCulate:MATH:STATe

Enables or disables the math expression.

Syntax

:CALCulate[*c*]:MATH:STATe *mode*

:CALCulate[*c*]:MATH:STATe?

Parameter

mode 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the math expression.

mode=0 or OFF disables the math expression.

Query response

mode <newline>

mode is 0 or 1, and indicates that the math expression is off or on, respectively. Response data type is NR1.

Example

:CALC:MATH:STAT 1

:CALC2:MATH:STAT?

:CALCulate:MATH:UNITs

Defines the unit name for the math expression.

Syntax

:CALCulate[*c*]:MATH:UNITs *unit*

:CALCulate[*c*]:MATH:UNITs?

Parameter

unit Unit name. Up to 32 ASCII characters. Parameter data type is SPD.

Query response

unit <newline>

unit returns the unit name of the math expression. Response data type is SRD.

Example

:CALC:MATH:UNIT “amps”

:CALC2:MATH:UNIT?

:CALCulate:OFFSet

Sets the *null offset* value used for calculating the limit test data.

The null offset function is enabled by the :CALCulate:OFFSet:STATe command.

Syntax

:CALCulate[*c*]:OFFSet *offset*

:CALCulate[*c*]:OFFSet? [*offset*]

Parameter

offset *value* (-9.999999E+20 to +9.999999E+20)|MINimum|MAXimum|DEFault (default is 0.0). Parameter data type is NRf+. Query does not support *offset=value*.

Query response

offset <newline>

offset returns the present setting of the null offset value. If a parameter is specified, *offset* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:CALC:OFFS 0.5

:CALC2:OFFS?

:CALCulate:OFFSet:ACQuire

Automatically sets the *null offset* value used for calculating the limit test data. The *null offset* value will be the currently available value read by the :CALCulate:DATA:LATest? or :SENSe:DATA:LATest? command. Or, it will be 0.0 if a currently available value does not exist.

Syntax :CALCulate[*c*]:OFFSet:ACQuire

Example :CALC:OFFS:ACQ

:CALC2:OFFS:ACQ

:CALCulate:OFFSet:STATe

Enables or disables the null offset function used for calculating the limit test data.

The null offset value is set by the :CALCulate:OFFSet or :CALCulate:OFFSet:ACQuire command.

Syntax :CALCulate[*c*]:OFFSet:STATe *mode*

:CALCulate[*c*]:OFFSet:STATe?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the null offset.

mode=0 or OFF disables the null offset.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the null offset is off or on, respectively. Response data type is NR1.

Example :CALC:OFFS:STAT 1

:CALC2:OFFS:STAT?

DISPlay Subsystem

For the numeric suffix [*d*], see “Numeric Suffix” on page 1-8.

:DISPlay:CSET

Selects the color set of the front panel display. This command setting is not changed by power off or the *RST command.

Syntax

:DISPlay:CSET *color*

:DISPlay:CSET?

Parameter

color Color set of the front panel display. 1|2. Parameter data type is NR1.

color=1 selects the default color set.

color=2 selects the alternative color set.

Query response

response <newline>

response is 1 or 2, and indicates the color set of the front panel display. Response data type is NR1.

Example

:DISP:CSET 1

:DISP:CSET?

:DISPlay:DIGits

Sets the display resolution of the data displayed on the front panel display.

Syntax

:DISPlay:DIGits *digits*

:DISPlay:DIGits? [MINimum|MAXimum|DEFault]

Parameter

digits Resolution. *value* (4 to 7)|MINimum|MAXimum|DEFault (default is 7). Parameter data type is NRf+. Query does not support *digits=value*.

digits=4 selects 3½ digit resolution.

digits=5 selects 4½ digit resolution.

digits=6 selects 5½ digit resolution.

digits=7 selects 6½ digit resolution.

Query response *digits* <newline>

digits returns the present setting. If a parameter is specified, *digits* returns the value assigned to DEF, MIN, or MAX. Response data type is NR1.

Example :DISP:DIG 4

:DISP:DIG? MAX

:DISPlay:ENABle

Enables or disables the front panel display under remote operation. This command setting is not changed by power off or the *RST command.

Regardless of this command setting, the front panel keys and the rotary knob are disabled during remote operation. However, only the *Local* key is effective for returning the instrument to local operation.

Syntax :DISPlay:ENABle *mode*

:DISPlay:ENABle?

Parameter ***mode*** 0|OFF|1|ON. Parameter data type is boolean.

mode=1 or ON enables the front panel display.

mode=0 or OFF disables the front panel display.

Query response *mode* <newline>

mode is 0 or 1, and indicates that front panel display is off or on, respectively. Response data type is NR1.

Example :DISP:ENAB OFF

:DISP:ENAB?

:DISPlay:VIEW

Sets the display mode, single 1, single 2, dual, graph, or roll.

Syntax :DISPlay:VIEW *mode*

:DISPlay:VIEW?

Parameter

mode SINGle1|SINGle2|DUAL|GRAPh|ROLL. Parameter data type is CPD.

mode=SINGle1 sets the channel 1 display mode. Default setting for 1-channel models.

mode=SINGle2 sets the channel 2 display mode. Only on 2-channel models.

mode=DUAL sets channel 1, and channel 2 display mode. Only on 2-channel models. Default setting.

mode=GRAPh sets the graph display mode for the sweep measurement results.

mode=ROLL sets the roll display mode for the time domain measurement results.

Query response

mode <newline>

mode returns SING1, SING2, DUAL, GRAP, or ROLL. Response data type is CRD.

Example

:DISP:VIEW GRAP

:DISP:VIEW?

:DISPlay[:WINDow]:DATA?

Returns the data displayed on the front panel display.

Syntax

:DISPlay[:WINDow[*d*]]:DATA?

Query response

response <newline>

response returns the measured values, source output value, and sense limit value displayed on the front panel display, as shown below. Each data is separated by a comma. *response* returns ---------- (hyphen) for the empty data. Response data type is SRD.

meas_value1,*meas_value2*,*source_output*,*sense_limit*

Characters μ and Ω are converted to u and ohm, respectively.

For the SING1, SING2, or DUAL display mode, the :DISP:DATA? command returns the data displayed on the upper display area.

For the SING1 or SING2 display mode, the :DISP:WIND2:DATA? command returns “----------,----------,----------,----------”.

For the DUAL display mode, the :DISP:WIND2:DATA? command returns the data displayed on the lower display area.

For the GRAPh or ROLL display mode, *response* returns "----------,----------,----------,----------".

Example :DISP:DATA?

:DISP:WIND2:DATA?

:DISPlay[:WINDow]:TEXT:DATA

Sets the text message displayed on the center of the upper or lower display area of the front panel display.

Syntax :DISPlay[:WINDow[*d*]]:TEXT:DATA *text*

:DISPlay[:WINDow[*d*]]:TEXT:DATA?

Parameter ***text*** Text. Up to 32 ASCII characters. Parameter data type is SPD.

Query response *text* <newline>

text returns the text message. Response data type is SRD.

Example :DISP:TEXT:DATA "Sweep measurement"

:DISPlay[:WINDow]:TEXT:STATe

Shows or hides the text message set by the :DISPlay[:WINDow]:TEXT:DATA command.

Syntax :DISPlay[:WINDow[*d*]]:TEXT:STATe *mode*

:DISPlay[:WINDow[*d*]]:TEXT:STATe?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON shows the text message.

mode=0 or OFF hides the text message.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the text message is off or on, respectively. Response data type is NR1.

Example :DISP:TEXT:STAT 1

:DISPlay:ZOOM

Enables or disables the zoom function of the front panel display. This function is effective for the Dual view and Single view.

If this function is enabled, the setup information is not displayed and the measurement result is zoomed. Then,

- the Dual view displays the primary measurement data with a large font, and the secondary measurement data with a small font, for each channel.
- the Single view displays both primary and secondary measurement data with a large font.

Syntax :DISPlay:ZOOM *mode*

:DISPlay:ZOOM?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the zoom function of the front panel display.

mode=0 or OFF disables the zoom function of the front panel display.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the front panel zoom function is off or on, respectively. Response data type is NR1.

Example :DISP:ZOOM ON

:DISP:ZOOM?

FETCh Subsystem

:FETCh:ARRay?

Returns the array data which contains all of the voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:FETCh:ARRay? [*chanlist*]

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the array data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr1,*ch1sour1*,*ch2curr1*,*ch2sour1*,*ch1curr2*,*ch1sour2*,*ch2curr2*,*ch2sour2*, ...
ch1curr5,*ch1sour5*,*ch2curr5*,*ch2sour5*,*ch1curr6*,*ch1sour6*,+9.910000E+37,+9.910000E+37, ...
ch1curr10,*ch1sour10*,+9.910000E+37,+9.910000E+37

This example shows the data containing the current data (*ch1currN*) and source data (*ch1sourN*) of the 10-step sweep measurement by channel 1, and the current data (*ch2currN*) and source data (*ch2sourN*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:FETC:ARR? (@1,2)
```

:FETCh:ARRay:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>?

Returns the array data which contains all of the current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:FETCh:ARRay:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? [*chanlist*]

For <CURRent|RESistance|SOURce|STATus|TIME|VOLTage>, specify CURRent for current measurement data, RESistance for resistance measurement data, SOURce for source output setting data, STATus for status data, TIME for time data, or VOLTage for voltage measurement data.

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the array data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr1,*ch2curr1*,*ch1curr2*,*ch2curr2*, ...
ch1curr5,*ch2curr5*,*ch1curr6*,+9.910000E+37, ...
ch1curr10,+9.910000E+37

This example shows the data containing the current data (*ch1currN*) of the 10-step sweep measurement by channel 1, and the current data (*ch2currN*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

:FETC:ARR:CURR? (@2,1)

:FETCh[:SCALar]?

Returns the latest voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:FETCh[:SCALar]? [*chanlist*]

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the latest data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr10,ch1sour10,ch2curr5,ch2sour5

This example shows the data containing the latest current data (*ch1curr10*) and source data (*ch1sour10*) of the 10-step sweep measurement by channel 1, and the latest current data (*ch2curr5*) and source data (*ch2sour5*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:FETC? (@1,2)
```

:FETCh[:SCALar]:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>?

Returns the latest current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:FETCh[:SCALar]:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? [*chanlist*]

For <CURRent|RESistance|SOURce|STATus|TIME|VOLTage>, specify CURRent for current measurement data, RESistance for resistance measurement data, SOURce for source output setting data, STATus for status data, TIME for time data, or VOLTage for voltage measurement data.

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the latest data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr10,ch2curr5

This example shows the data containing the latest current data (*ch1curr10*) of the 10-step sweep measurement by channel 1, and the latest current data (*ch2curr5*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example :FETC:CURR? (@2,1)

FORMat Subsystem

:FORMat:BORDer

This command is effective when the data output format is set to the IEEE-754 binary format by using the :FORMat[:DATA] command. This command sets the byte order of binary output data.

Syntax

:FORMat:BORDer *byte_order*

:FORMat:BORDer?

Parameter

byte_order NORMal (default)|SWAPped. Parameter data type is CPD.

byte_order=NORMal sets the normal byte order. For the IEEE-754 single precision format, byte 1 to byte 4 are sent in this order. For the IEEE-754 double precision format, byte 1 to byte 8 are sent in this order.

byte_order=SWAPped sets the reverse byte order. For the IEEE-754 single precision format, byte 4 to byte 1 are sent in this order. For the IEEE-754 double precision format, byte 8 to byte 1 are sent in this order.

Query response

byte_order <newline>

byte_order returns NORM or SWAP. Response data type is CRD.

Example

:FORM:BORD SWAP

:FORM:BORD?

:FORMat[:DATA]

Sets the data output format. See “Data Output Format” on page 1-12.

Syntax

:FORMat [:DATA] *format*

:FORMat [:DATA]?

Parameter

format Data output format. Parameter data type is CPD.

ASCii|REAL,32|REAL,64. RERL,64 is only for the default language mode set by the :SYST:LANG “DEF” command.

format=ASCii specifies the ASCII format (default).

format=REAL,32 specifies the IEEE-754 single precision format. 4-byte data.

format=REAL,64 specifies the IEEE-754 double precision format. 8-byte data.

Query response *format* <newline>

format returns ASC, REAL,32, or REAL,64. Response data type is CRD.

Example :FORM REAL32

:FORM?

:FORMat:DIGital

Sets the response format of the bit pattern defined by the following commands.

- :CALCulate:CLIMits:<FAIL|PASS>:DIGital[:DATA]
- :CALCulate:LIMit:COMPliance:DIGital[:DATA]
- :CALCulate:LIMit:<LOWer|UPPer>:DIGital[:DATA]
- :CALCulate:LIMit:PASS:DIGital[:DATA]

Syntax :FORMat:DIGital *format*

:FORMat:DIGital?

Parameter

format	Response format. ASCii (decimal, default)| BINary|OCTal|HEXadecimal. Parameter data type is CPD.

Query response *format* <newline>

format returns ASC, BIN, OCT, or HEX. Response data type is CRD.

Example :FORM:DIG BIN

:FORM:DIG?

:FORMat:ELEMents:CALCulate

Specifies the elements included in the calculation result data returned by the :CALCulate:DATA?, :CALCulate:DATA:LATest?, :CALCulate:MATH:DATA?, :CALCulate:MATH:DATA:LATest?, or :TRACe:DATA? command.

For the data stored in the trace buffer, this command is effective for the calculation result data or limit test data that is specified by the :TRACe:FEED MATH|LIM command.

If all elements are specified by this command, the result data contains the all elements shown below. Then the order of elements is exclusive. For example, if TIME is not specified, the data contains the *calc* and *status* data in this order. If this command is not entered, the data contains the *calc* data only.

Elements and their order: *calc*, *time*, *status*

Syntax

:FORMat:ELEMents:CALCulate *type*{,*type*}

:FORMat:ELEMents:CALCulate?

Parameter

type Data element included in the data. CALC (calculation data, default)|TIME|STATus. Parameter data type is CPD.

CALC selects the calculation data *calc*.

TIME selects the time (timestamp) data *time*.

STAT selects the status data *status*.

Query response

type{,*type*} <newline>

type returns CALC, TIME, or STAT. Response data type is CRD. Multiple responses are separated by a comma.

Example

:FORM:ELEM:CALC CALC,TIME,STAT

:FORM:ELEM:CALC?

:FORMat:ELEMents:SENSe

Specifies the elements included in the sense or measurement result data returned by the :FETCh?, :READ?, :MEASure?, or :TRACe:DATA? command.

For the data stored in the trace buffer, this command is effective for the measurement result data that is specified by the :TRACe:FEED SENS command.

If this command is not entered or if all elements are specified by this command, the sense or measurement result data contains the all elements shown below. Then the order of elements is exclusive. For example, if VOLTage and RESistance are not specified, the data contains the *current*, *time*, *status*, and *source* data in this order. It will not contain the *voltage* and *resistance* data.

Elements and their order: *voltage*, *current*, *resistance*, *time*, *status*, *source*

Syntax

:FORMat:ELEMents:SENSe *type*{,*type*}

:FORMat:ELEMents:SENSe?

Parameter

type	Data element included in the data. VOLTage|CURRent|RESistance|TIME|STATus|SOURce. Parameter data type is CPD.

VOLT selects the voltage measurement data *voltage*.

CURR selects the current measurement data *current*.

RES selects the resistance measurement data *resistance*.

TIME selects the time data *time* (timestamp of the measurement start trigger).

STAT selects the status data *status*.

SOUR selects the source output setting data *source*.

Query response

type{,*type*} <newline>

type returns VOLT, CURR, RES, TIME, STAT, or SOUR. Response data type is CRD. Multiple responses are separated by a comma.

Example

:FORM:ELEM:SENS SOUR,CURR,VOLT,RES,TIME,STAT

:FORM:ELEM:SENS?

:FORMat:SREGister

Sets the response format of the status byte register.

Syntax

:FORMat:SREGister *format*

:FORMat:SREGister?

Parameter

format	Response format. ASCii (decimal, default)| BINary|OCTal|HEXadecimal. Parameter data type is CPD.

Query response

format <newline>

format returns ASC, BIN, OCT, or HEX. Response data type is CRD.

Example

:FORM:SREG BIN

:FORM:SREG?

HCOPy Subsystem

:HCOPy:SDUMp:DATA?

Returns the data of the front panel screen image. The format of the image data is set by the :HCOPy:SDUMp:FORMat command.

Syntax :HCOPy:SDUMp:DATA?

Query response The response is a definite length arbitrary binary block.

Example :HCOP:SDUM:DATA?

:HCOPy:SDUMp:FORMat

Sets the format of the image data. The front panel screen image will be created in the format set by this command. The image data will be returned by the :HCOPy:SDUMp:DATA? command.

Syntax :HCOPy:SDUMp:FORMat *format*

HCOPy:SDUMp:FORMat?

Parameter

format	Format of image data. JPG (default)|BMP|PNG|WMF. Parameter data type is CPD.

Query response *format* <newline>

format returns JPG, BMP, PNG, or WMF. Response data type is CRD.

Example :HCOP:SDUM:FORM BMP

:HCOP:SDUM:FORM?

LXI Subsystem

:LXI:IDENtify[:STATe]

Changes the LXI status indicator state.

Syntax

:LXI:IDENtify[:STATe] *mode*

:LXI:IDENtify[:STATe]?

Parameter

mode 0|OFF (default)|1|ON . Parameter data type is boolean.

mode=1 or ON changes the LXI status indicator to the Identify state.

mode=0 or OFF changes the LXI status indicator to the No Fault state.

Query response

mode <newline>

mode returns 0 or 1, and indicates that the LXI status indicator is No Fault or Identify, respectively. Response data type is NR1.

Example

:LXI:IDEN 0

:LXI:IDEN:STAT?

:LXI:MDNS:ENABle

Enables or disables mDNS (multicast DNS) function.

Syntax

:LXI:MDNS:ENABle *mode*

:LXI:MDNS:ENABle?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the mDNS function.

mode=0 or OFF disables the mDNS function.

Query response

mode <newline>

mode returns 0 or 1, and indicates that the mDNS function is disable or enable, respectively. Response data type is NR1.

Example :LXI:MDNS:ENAB 0

:LXI:MDNS:ENAB?

:LXI:MDNS:HNAMe[:RESolved]?

Returns the resolved mDNS hostname.

Syntax :LXI:MDNS:HNAMe[:RESolved]?

Query response *desired mDNS hostname - N* <newline>

N is an integer appended as necessary to make the name unique. Response data type is SRD.

Example :LXI:MDNS:HNAM?

:LXI:MDNS:SNAMe:DESired

Sets the desired mDNS service name.

Syntax :LXI:MDNS:SNAMe:DESired *name*

:LXI:MDNS:SNAMe:DESired?

Parameter

name	Desired mDNS service name. Up to 15 ASCII characters. Parameter data type is SPD.

Query response *name* <newline>

name returns the desired mDNS service name. Response data type is SRD.

Example :LXI:MDNS:SNAM:DES “B2900”

:LXI:MDNS:SNAM:DES?

:LXI:MDNS:SNAMe[:RESolved]?

Returns the resolved mDNS service name.

Syntax :LXI:MDNS:SNAMe[:RESolved]?

Query response *desired mDNS service name - N* <newline>

N is an integer appended as necessary to make the name unique. Response data type is SRD.

Example :LXI:MDNS:SNAM?

Other LXI Subsystem Commands

Agilent B2900 also supports the following commands. For details, refer to SCPI documents.

- :ARM:LXI:COUNt
- :ARM:LXI:COUNt?
- :ARM:LXI:DELay
- :ARM:LXI:DELay?
- :ARM:LXI:LAN[:SET]:DETection
- :ARM:LXI:LAN[:SET]:DETection?
- :ARM:LXI:LAN[:SET]:ENABle
- :ARM:LXI:LAN[:SET]:ENABle?
- :ARM:LXI:LAN[:SET]:FILTer
- :ARM:LXI:LAN[:SET]:FILTer?
- :ARM:LXI:LAN[:SET]:IDENtifier
- :ARM:LXI:LAN[:SET]:IDENtifier?
- :LXI:EVENt:DOMain
- :LXI:EVENt:DOMain?
- :LXI:EVENt:INPut:LAN[:SET]:DELay
- :LXI:EVENt:INPut:LAN[:SET]:DELay?
- :LXI:EVENt:INPut:LAN[:SET]:DETection
- :LXI:EVENt:INPut:LAN[:SET]:DETection?
- :LXI:EVENt:INPut:LAN[:SET]:ENABle
- :LXI:EVENt:INPut:LAN[:SET]:ENABle?
- :LXI:EVENt:INPut:LAN[:SET]:FILTer
- :LXI:EVENt:INPut:LAN[:SET]:FILTer?

- :LXI:EVENt:INPut:LAN[:SET]:IDENtifier
- :LXI:EVENt:INPut:LAN[:SET]:IDENtifier?
- :LXI:EVENt:LOG:CIRCular[:ENABle]
- :LXI:EVENt:LOG:CIRCular[:ENABle]?
- :LXI:EVENt:LOG:ENABle
- :LXI:EVENt:LOG:ENABle?
- :LXI:EVENt:LOG:SIZE
- :LXI:EVENt:LOG:SIZE?
- :LXI:EVENt:STATus:ENABle
- :LXI:EVENt:STATus:ENABle?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:DESTination
- :LXI:EVENt[:OUTPut]:LAN[:SET]:DESTination?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:DRIVe
- :LXI:EVENt[:OUTPut]:LAN[:SET]:DRIVe?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:ENABle
- :LXI:EVENt[:OUTPut]:LAN[:SET]:ENABle?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:IDENtifier
- :LXI:EVENt[:OUTPut]:LAN[:SET]:IDENtifier?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:SLOPe
- :LXI:EVENt[:OUTPut]:LAN[:SET]:SLOPe?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:SOURce
- :LXI:EVENt[:OUTPut]:LAN[:SET]:SOURce?
- :LXI:EVENt[:OUTPut]:LAN[:SET]:TSDelta
- :LXI:EVENt[:OUTPut]:LAN[:SET]:TSDelta?
- :TRIGger:LXI:LAN[:SET]:DELay
- :TRIGger:LXI:LAN[:SET]:DELay?
- :TRIGger:LXI:LAN[:SET]:DETection
- :TRIGger:LXI:LAN[:SET]:DETection?

- :TRIGger:LXI:LAN[:SET]:ENABle
- :TRIGger:LXI:LAN[:SET]:ENABle?
- :TRIGger:LXI:LAN[:SET]:FILTer
- :TRIGger:LXI:LAN[:SET]:FILTer?
- :TRIGger:LXI:LAN[:SET]:IDENtifier
- :TRIGger:LXI:LAN[:SET]:IDENtifier?

LXI Trigger Events

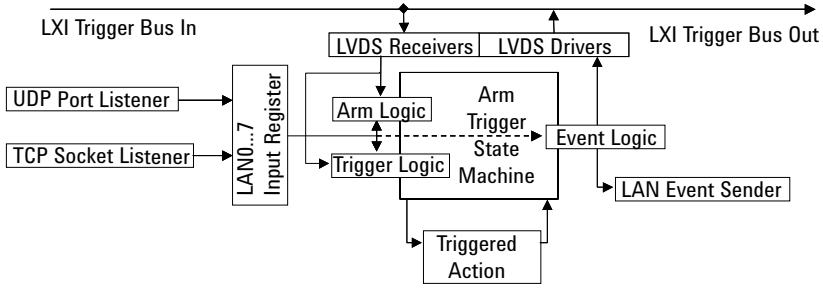
B2900 provides a subset of the LXI Trigger Events (IVI-3.15 IviLxiSync) functionality in the trigger system.

- Device Model

  Figure 4-1 shows the high-level LXI device model defined in IVI-3.15. B2900 does not have the LXI Trigger Bus, but has the UDP Port/TCP Socket Listener and the LAN Event Sender in the system.

  You can configure the trigger systems to send/receive LAN*n* (*n*: 0 to 7) trigger events by the instrument specific trigger event, slope, drive logic, destination, and filter.

Figure 4-1 High-Level LXI Device Model



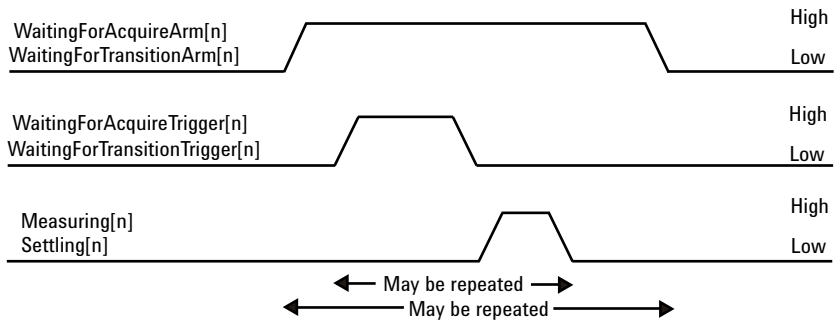
- Instrument Specific Events

  B2900 has the ARM-TRIGger model for each channel and actions (transition and acquire), and provides following events.

  - WaitingForAcquireArm1, WaitingForAcquireArm2
  - WaitingForAcquireTrigger1, WaitingForAcquireTrigger2
  - WaitingForTransitionArm1, WaitingForTransitionArm2
  - WaitingForTransitionTrigger1, WaitingForTransitionTrigger2
  - Measuring1, Measuring2
  - Settling1, Settling2

  All events can be configured by the signal level (or edge), destination, and other parameters defined in IVI 3.15.

Figure 4-2 Trigger State Machine Signal Relationships



- Limitations

  LXI trigger event functions provided by B2900 are a subset of the IEEE-1588 required by LXI Class-B. The following limitations exist for B2900.

  - Timestamp in the event are ignored. (immediate trigger only)
  - Delay and other timing parameters cannot be set. (always 0)
  - It is not allowed to add/delete any events.
  - The :ARM:LXI:COUNt command is not effective. (ignored)

MEASure Subsystem

:MEASure?

Executes a spot measurement (one-shot measurement) and returns the measurement result data. Measurement conditions must be set by SCPI commands or front panel operation before executing this command. Measurement items can be selected by the :FORMat:ELEMents:SENSe command.

Syntax

:MEASure? [*chanlist*]

Parameter

chanlist Channels to perform measurement. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See "Channel List Parameter" on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the measurement result data. Response data type is NR3. See "Data Output Format" on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr,ch1sour,ch2curr,ch2sour

This example shows the data containing the current data (*ch1curr*) and source data (*ch1sour*) of channel 1, and the current data (*ch2curr*) and source data (*ch2sour*) of channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates "not a number".

Example

```
:FORM:ELEM:SENS CURR,SOUR
:MEAS? (@1,2)
```

:MEASure:<CURRent|RESistance|VOLTage>?

Executes a spot measurement (one-shot measurement) and returns the measurement result data. Measurement conditions must be set by SCPI commands or front panel operation before executing this command. Measurement item can be set to CURRent, RESistance, or VOLTage.

Syntax

:MEASure:<CURRent[:DC]|RESistance|VOLTage[:DC]>? [*chanlist*]

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, select CURRent[:DC] for current measurement, RESistance for resistance measurement, or VOLTage[:DC] for voltage measurement.

Parameter

chanlist Channels to perform measurement. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the measurement result data. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr,ch2curr

This example shows the data containing the current data (*ch1curr*) of channel 1, and the current data (*ch2curr*) of channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

:MEAS:CURR? (@2,1)

MMEMory Subsystem

:MMEMory:CATalog?

Returns the memory usage and availability. Also returns the list of files and folders in the current specified directory.

Syntax

:MMEMory:CATalog? [*directory*]

Parameter

directory Directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator. Up to 255 ASCII characters. Parameter data type is SPD.

If *directory* is not set, this function is applied to the current directory.

If *directory*=<path>, this function is applied to <current directory>\<path>.

If *directory*=USB:\<path>, this function is applied to USB:\<path>. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Error occurs if the specified directory does not exist or is set to hidden or system.

Query response

used,*free*{,*item*}<newline>

used returns the size of the used space, in bytes. Response data type is NR1.

free returns the size of the free space, in bytes. Response data type is NR1.

item returns the file or directory information. Response data type is SRD.

For files, *item* returns a string *name*,*type*,*size* which indicates the file name, file type, and file size. Where, *type* returns “ASC”, “BIN”, “STAT” or “MACR” for the file extensions “csv”, “dat”, “sta”, and “mac”, respectively.

For a directory, *item* returns a string *name*,*type*,*size*. Where, *name* indicates the directory name, and *type*,*size* always returns “FOLD,0”.

Example

:MMEM:CAT? “USB:\b2900\device1\iv_test\result”

:MMEM:CAT? “b2900\device1\iv_test\result”

:MMEMory:CDIRectory

Changes the current directory to the specified directory.

Syntax

:MMEMory:CDIRectory *directory*

:MMEMory:CDIRectory?

Parameter

directory Directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator. Up to 255 ASCII characters. Parameter data type is SPD.

If *directory*=<path>, the next current directory will be <current directory>\<path>.

If *directory*=USB:\<path>, the next current directory will be USB:\<path>. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Error occurs if the specified directory does not exist or is set to hidden or system.

Query response

directory <newline>

directory returns the full path of the current directory. Response data type is SRD.

Example

:MMEM:CDIR “USB:\b2900\device1\iv_test\result”

:MMEM:CDIR?

:MMEMory:COPY

Makes a copy of an existing file in the current directory.

Syntax

:MMEMory:COPY *source*,*destination*

Parameter

source Source file name. Name of the original file.

destination Copy file name. Or directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator.

Length of parameters is up to 255 ASCII characters. Parameter data type is SPD.

If *destination* is a file name, the copy file is created in the current directory.

If *destination*=<path>, the source file is duplicated in <current directory>\<path>.

If *destination*=USB:\<path>, the source file is duplicated in USB:\<path>. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Error occurs if the source file does not exist or the destination file already exists.

Example

:MMEM:COPY “original.dat”,“original_copy.dat”

:MMEM:COPY “original.dat”,“USB:\b2900\device1\iv_test\result”

:MMEMory:DELete

Deletes a file in the current directory.

Syntax :MMEMory:DELete *file_name*

Parameter

file_name	Name of the file to delete. Up to 255 ASCII characters. Parameter data type is SPD.

Error occurs if the specified file does not exist.

Example :MMEM:DEL “original_copy.dat”

:MMEMory:LOAD:MACRo

Loads a macro from the specified file in the current directory.

Syntax :MMEMory:LOAD:MACRo *macro,file_name*

Parameter

macro	Name of macro.
file_name	Name of the file which contains the macro. File extention must be *mac*.

Length of parameters is up to 255 ASCII characters. Parameter data type is SPD.

Example :MMEM:LOAD:MACR “abc”,“MacroData1.mac”

:MMEMory:LOAD:STATe

Loads an instrument setup from the specified file in the current directory.

Syntax :MMEMory:LOAD:STATe *file_name*

Parameter

file_name	Name of the file which contains the instrument setup. File extention must be *sta*. Up to 255 ASCII characters. Parameter data type is SPD.

Example :MMEM:LOAD:STAT “SetupData1.sta”

:MMEMory:MDIRectory

Creates a new directory.

Syntax :MMEMory:MDIRectory *directory*

Parameter

directory Directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator. Up to 255 ASCII characters. Parameter data type is SPD.

If *directory*=<path>, this command creates a <current directory>\<path> directory.

If *directory*=USB:\<path>, this command creates a USB:\<path> directory. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Example :MMEM:MDIR “USB:\b2900\device1\iv_test\setup”

:MMEMory:MOVE

Moves or renames an existing file in the current directory.

Syntax :MMEMory:MOVE *source*,*destination*

Parameter

source Source file name. Name of the original file.

destination New file name. Or directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator.

Length of parameters is up to 255 ASCII characters. Parameter data type is SPD.

If *destination* is a file name, the source file is renamed to the new file name in the current directory.

If *destination*=<path>, the source file is moved to <current directory>\<path>.

If *destination*=USB:\<path>, the source file is moved to USB:\<path>. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Error occurs if the source file does not exist or the destination file already exists.

Example :MMEM:MOVE “original.dat”,“new.dat”

:MMEM:MOVE “original.dat”,“USB:\b2900\device1\iv_test\result”

:MMEMory:RDIRectory

Removes the specified empty directory.

Syntax :MMEMory:RDIRectory *directory*

Parameter

directory Directory name, <path>|USB:\<path>. Either / (slash) or \ (backslash) can be used as the path separator. Up to 255 ASCII characters. Parameter data type is SPD.

If *directory*=<path>, this command removes the <current directory>\<path> directory.

If *directory*=USB:\<path>, this command removes the USB:\<path> directory. Where, USB:\ is the root directory of the USB memory connected to the B2900 front panel.

Error occurs if the specified directory is not empty.

Example

:MMEM:RDIR “USB:\b2900\device1\iv_test\setup”

:MMEMory:STORe:DATA<:LIMit|:MATH|:SENSe|[:ALL]>

Saves the limit test data, math expression result data, sense data, or all of these data for the specified channel to the specified file in the current directory.

Syntax

:MMEMory:STORe:DATA<:LIMit|:MATH|:SENSe|[:ALL]> *file_name*[,*chanlist*]

For <:LIMit|:MATH|:SENSe|[:ALL]>, specify :LIMit for limit test data, :MATH for math expression result data, :SENSe for sense data, or [:ALL] for all of these data.

Parameter

file_name Name of the file used to save the specified data. Up to 255 ASCII characters. Parameter data type is SPD. File extention must be *dat*, which is meaningful for the :MMEMory:CATalog? result.

chanlist Channels to collect the data for. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If *chanlist* is not specified, *chanlist*=(@1) is set.

Example

:MMEM:STOR:DATA “AllData1.dat”

:MMEMory:STORe:MACRo

Saves the macro to the specified file in the current directory.

Syntax :MMEMory:STORe:MACRo *macro*,*file_name*

Parameter

macro	Name of macro.
file_name	Name of the file used to save the macro. File extention must be *mac*, which is meaningful for the :MMEMory:CATalog? result.

Length of parameters is up to 255 ASCII characters. Parameter data type is SPD.

Example :MMEM:STOR:MACR “abc”,“MacroData1.mac”

:MMEMory:STORe:STATe

Saves the instrument setup to the specified file in the current directory.

Syntax :MMEMory:STORe:STATe *file_name*

Parameter

file_name	Name of the file used to save the instrument setup. Up to 255 ASCII characters. Parameter data type is SPD. File extention must be *sta*, which is meaningful for the :MMEMory:CATalog? result.

Example :MMEM:STOR:STAT “SetupData1.sta”

:MMEMory:STORe:TRACe

Saves all data in the trace buffer for the specified channel to the specified file in the current directory.

Syntax :MMEMory:STORe:TRACe *file_name*[,*chanlist*]

Parameter

file_name	Name of the file used to save the specified data. Up to 255 ASCII characters. Parameter data type is SPD. File extention must be *tra*, which is meaningful for the :MMEMory:CATalog? result.
chanlist	Channels to get the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If *chanlist* is not specified, *chanlist*=(@1) is set.

Example :MMEM:STOR:TRAC “AllTraceData1.dat”

OUTPut Subsystem

For the numeric suffix [*c*], see “Numeric Suffix” on page 1-8.

:OUTPut:FILTer:AUTO

Enables or disables the automatic filter function.

Syntax

:OUTPut[*c*]:FILTer:AUTO *mode*

:OUTPut[*c*]:FILTer:AUTO?

Parameter

mode 1|ON|0|OFF (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic filter function.

mode=1 or ON enables the automatic filter function. If this function is enabled, the instrument automatically sets the output filter which provides the best filter characteristics and cutoff frequency. The following command settings are ignored.

- :OUTPut:FILTer[:LPASs]:FREQuency
- :OUTPut:FILTer[:LPASs]:TCONstant

Query response

mode <newline>

mode is 0 or 1, and indicates that the automatic filter function is off or on, respectively. Response data type is NR1.

Example

:OUTP:FILT:AUTO 1

:OUTP2:FILT:AUTO?

:OUTPut:FILTer[:LPASs]:FREQuency

Sets the cutoff frequency of the output filter. This command setting is ignored if the automatic filter function is enabled by the :OUTPut:FILTer:AUTO command.

Syntax

:OUTPut[*c*]:FILTer[:LPASs]:FREQuency *frequency*

:OUTPut[*c*]:FILTer[:LPASs]:FREQuency? [*frequency*]

Parameter

frequency *value* (31.830 Hz to +31.831 kHz)|MINimum|MAXimum|DEFault. Parameter data type is NRf+. Query does not support *frequency=value*. If you specify the value less than MIN or greater than MAX, *frequency* is automatically set to MIN or MAX.

The cutoff frequency can be expressed by the following formula, using the time constant set by the :OUTPut:FILTer[:LPASs]:TCONstant command. So the last command setting is effective for both *frequency* and *time_constant*.

$frequency = 1/(2 \times \pi \times time\_constant)$

Query response

frequency <newline>

frequency returns the present setting of the cutoff frequency. If a parameter is specified, *frequency* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:OUTP:FILT:FREQ 1E4

:OUTP2:FILT:LPAS:FREQ?

:OUTPut:FILTer[:LPASs][:STATe]

Enables or disables the output filter.

Syntax

:OUTPut[*c*]:FILTer[:LPASs][:STATe] *mode*

:OUTPut[*c*]:FILTer[:LPASs][:STATe]?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the output filter.

mode=0 or OFF disables the output filter.

Query response

mode <newline>

mode is 0 or 1, and indicates that the output filter is off or on, respectively. Response data type is NR1.

Example

:OUTP:FILT 0

:OUTP2:FILT:LPAS:STAT?

:OUTPut:FILTer[:LPASs]:TCONstant

Sets the time constant instead of setting the cutoff frequency of the output filter. This command setting is ignored if the automatic filter function is enabled by the :OUTPut:FILTer:AUTO command.

Syntax

:OUTPut[*c*]:FILTer[:LPASs]:TCONstant *time_constant*

:OUTPut[*c*]:FILTer[:LPASs]:TCONstant? [*time_constant*]

Parameter

time_constant *value* (5 µs to 5 ms)|MINimum|MAXimum| DEFault. Parameter data type is NRf+. Query does not support *time_constant=value*. If you specify the value less than MIN or greater than MAX, *time* is automatically set to MIN or MAX.

The time constant can be expressed by the following formula, using the cutoff frequency set by the :OUTPut:FILTer[:LPASs]:FREQuency command. So the last command setting is effective for both *time_constant* and *frequency*.

time_constant= $1/(2 \times \pi \times$ *frequency*)

Query response

time_constant <newline>

time_constant returns the present setting of the cutoff frequency. If a parameter is specified, *time_constant* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:OUTP:FILT:TCON 1E-6

:OUTP2:FILT:LPAS:TCON?

:OUTPut:HCAPacitance[:STATe]

Enables or disables the high capacitance mode. This mode is effective for high capacitive DUT.

Syntax

:OUTPut[*c*]:HCAPacitance[:STATe] *mode*

:OUTPut[*c*]:HCAPacitance[:STATe]?

Parameter

mode 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the high capacitance mode.

mode=0 or OFF disables the high capacitance mode.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the high capacitance mode is off or on, respectively. Response data type is NR1.

Example :OUTP:HCAP 1

:OUTP2:HCAP:STAT?

:OUTPut:LOW

Selects the state of the low terminal. Before executing this command, the source output must be disabled by the :OUTPut[:STATe] command. Or else, an error occurs.

Syntax :OUTPut[*c*]:LOW *low_state*

:OUTPut[*c*]:LOW?

Parameter ***low_state*** FLOat|GROund (default). Parameter data type is CPD.

low_state=FLOat sets the floating state.

low_state=GROund sets the ground state. The low terminal is connected to ground.

Query response *low_state* <newline>

low_state is FLO or GRO, and indicates the low terminal state. Response data type is CRD.

Example :OUTP:LOW FLO

:OUTP2:LOW?

:OUTPut:OFF:AUTO

Enables or disables the automatic output off function.

Syntax :OUTPut[*c*]:OFF:AUTO *mode*

:OUTPut[*c*]:OFF:AUTO?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic output off function.

mode=1 or ON enables the automatic output off function. If this function is enabled, the source output is automatically turned off immediately when the grouped channels change status from busy to idle.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the automatic output off function is off or on, respectively. Response data type is NR1.

Example :OUTP:OFF:AUTO 1

:OUTP2:OFF:AUTO?

:OUTPut:OFF:MODE

Selects the source condition after output off.

Syntax :OUTPut[*c*]:OFF:MODE *mode*

:OUTPut[*c*]:OFF:MODE?

Parameter ***mode*** ZERO|HIZ|NORMal (default). Parameter data type is CPD.

mode=NORMal selects the following setup.

- Source function: Voltage source
- Output voltage: 0 V
- Current compliance: 100 μA at the 100 μA range
- Output relay: off (open or break)

mode=HIZ selects the following setup.

- Output relay: off (open or break)
- Voltage source setup is not changed if the source applies 40 V or less.
- Current source setup is not changed if the source uses the 100 mA range or lower.

mode=ZERO selects the following setup.

- Source function: Voltage source
- Output voltage: 0 V
- Current compliance: 100 μA at the 100 μA range

NOTE This command setting is not applied to the output-off process triggered by the emergency condition such as the over voltage/current protection, interlock open, and over temperature protection. Then the output voltage is immediately set to 0 V and the output switch is set to off.

Query response *mode* <newline>

mode is NORM, HIZ, or ZERO, and indicates the source condition after output off. Response data type is CRD.

Example :OUTP:OFF:MODE HIZ

:OUTP2:OFF:MODE?

:OUTPut:ON:AUTO

Enables or disables the automatic output on function.

Syntax :OUTPut[*c*]:ON:AUTO *mode*

:OUTPut[*c*]:ON:AUTO?

Parameter ***mode*** 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic output on function.

mode=1 or ON enables the automatic output on function. If this function is enabled, the source output is automatically turned on when the :INITiate or :READ command is sent.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the automatic output on function is off or on, respectively. Response data type is NR1.

Example :OUTP:ON:AUTO 0

:OUTP2:ON:AUTO?

:OUTPut:PROTection[:STATe]

Enables or disables the over voltage/current protection. If this function is enabled, the source/measure unit (SMU) sets the output to 0 V and sets the output switch to off automatically and immediately when it reaches the compliance status.

Syntax :OUTPut[*c*]:PROTection[:STATe] *mode*

:OUTPut[*c*]:PROTection[:STATe]?

Parameter ***mode*** 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the over voltage/current protection.

mode=1 or ON enables the over voltage/current protection.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the over voltage/current protection is off or on, respectively. Response data type is NR1.

Example :OUTP:PROT 0

:OUTP2:PROT:STAT?

:OUTPut:RECall

Recalls the channel setup saved by the :OUTPut:SAVE command.

Syntax :OUTPut[*c*]:RECall *index*

Parameter ***index*** 0|1. Parameter data type is NR1.

index=0 is used to recall channel setup 0.

index=1 is used to recall channel setup 1.

Example :OUTP:REC 1

:OUTPut:SAVE

Saves the channel setup. The setup can be recalled by the :OUTPut:RECall command.

Syntax :OUTPut[*c*]:SAVE *index*

Parameter ***index*** 0|1. Parameter data type is NR1.

index=0 is used to memorize the present channel setup as channel setup 0.

index=1 is used to memorize the present channel setup as channel setup 1.

Example :OUTP:SAVE 1

:OUTPut[:STATe]

Enables or disables the source output.

Syntax :OUTPut[*c*][:STATe] *mode*

:OUTPut[*c*][:STATe]?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the source output.

mode=0 or OFF disables the source output.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the source output is off or on, respectively. Response data type is NR1.

Example :OUTP 1

:OUTP2:STAT?

PROGram Subsystem

For the numeric suffix [*h*], see “Numeric Suffix” on page 1-8.

:PROGram:CATalog?

Returns the names of all programs defined in the program memory.

Even if a name is selected by the :PROGram[:SELected]:NAME command, this command does not return the name if the program is empty.

Syntax :PROGram:CATalog?

Query response *program_names* <newline>

program_names returns the names of all programs defined in the program memory. Response data type is AARD.

Example :PROG:CAT?

:PROGram:PON:COPY

Specifies the power-on program.

Syntax :PROGram:PON:COPY *name*

Parameter

name	Name of the program used for the power-on program. Parameter data type is SPD.

Example :PROG:PON:COPY “program1”

:PROGram:PON:DELete

Clears the power-on program.

Syntax :PROGram:PON:DELete

Example :PROG:PON:DEL

:PROGram:PON:RUN

Enables or disables the power-on program. The specified program automatically runs with each power-on. The program is specified by the :PROGram:PON:COPY command.

Syntax

:PROGram:PON:RUN *mode*

mode 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables power-on program.

mode=0 or OFF disables power-on program.

Query response

mode <newline>

mode is 0 or 1, and indicates that the power-on program is disable or enable, respectively. Response data type is NR1.

Example

:PROG:PON:RUN 1

:PROG:PON:RUN?

:PROGram[:SELected]:APPend

Adds a program code to the end of a program stored in the program memory.

Before executing this command, the program must be selected by the :PROGram[:SELected]:NAME command. Or else, an error occurs.

Syntax

:PROGram[:SELected]:APPend *program_code*

Parameter

program_code Program code. Up to 256 byte per execution. Sum of all program size in the program memory must be up to 100 KB. Parameter data type is block. Both definite length block and indefinite length block are available. Program code cannot contain control characters except for the trailing linefeed.

See the :PROGram[:SELected]:DEFine command for details.

Example

:PROG:NAME “program1”
:PROG:APP #213:OUTP:STAT ON

:PROGram[:SELected]:DEFine

Defines a program in the program memory by entering the initial program code.

Before executing this command, the program must be selected by the :PROGram[:SELected]:NAME command with a new program name. Or else, an error occurs.

Attempting to overwrite an existing program causes an error. Delete the program first by using the :PROGram[:SELected]:DELete[:SELected] command.

Syntax

:PROGram[:SELected]:DEFine *program_code*

:PROGram[:SELected]:DEFine?

Parameter

program_code Program code. Up to 256 byte per execution. Sum of all program size in the program memory must be up to 100 KB. Maximum of 100 programs can be memorized. Parameter data type is block. Both definite length block and indefinite length block are available. Program code cannot contain control characters except for the trailing linefeed.

For the definite length block, *program_code* must be #*nms* which consists of the header #*nm* and the command string *s*. For example, #213:OUTP:STAT ON.

- *n*: Number of digits for *m*. (ex: 2)
- *m*: Number of characters (8-bit data bytes) for the command string. (ex: 13)
- *s*: Command string. (ex: :OUTP:STAT ON (total 13 characters))

For the indefinite length block, *program_code* must be #0*s* which consists of the header #0 and the command string *s*. For example, #0:OUTP:STAT ON.

In the command string, the following characters have special meaning.

- \n: Command delimiter
- %%: Percent (%) character
- #: Comment header

program_code does not support the following.

- Query commands
- SCPI common commands except for *CLS, *ESE, and *SRE commands
- Program subsystem commands except for :PROG:STAT command

program_code supports variables in the format %*h*% (*h*: integer. 1 to 100). It is replaced with the value set by the :PROGram:VARiable command before executing the program.

Example

:PROG:NAME “program1”
:PROG:DEF #220:SOUR:FUNC:MODE CURR

:PROGram[:SELected]:DELete:ALL

Deletes all programs stored in the program memory.

If any of the programs are in the RUN state, this command causes an error and does not delete any program.

Syntax

:PROGram[:SELected]:DELete:ALL

Example

:PROG:DEL:ALL

:PROGram[:SELected]:DELete[:SELected]

Deletes a program stored in the program memory.

Before executing this command, the program must be selected by the :PROGram[:SELected]:NAME command. Or else, an error occurs.

If any of the programs are in the RUN state, this command causes an error and does not delete the selected program.

Syntax

:PROGram[:SELected]:DELete[:SELected]

Example

:PROG:NAME “program1”
:PROG:DEL

:PROGram[:SELected]:EXECute

Executes a program stored in the program memory.

Before executing this command, the program must be selected by the :PROGram[:SELected]:NAME command. Or else, an error occurs.

If any of the programs are in the RUN state, this command causes an error and does not execute the selected program.

Syntax

:PROGram[:SELected]:EXECute

Example

:PROG:NAME “program1”
:PROG:EXEC

:PROGram[:SELected]:NAME

Selects the program for performing the action by the following commands.

If *name* does not specify the program stored in the program memory, this command creates a new program with the specified name and selects the program.

If *name* specifies an existing program, this command selects the program.

Syntax

:PROGram[:SELected]:NAME *name*

:PROGram[:SELected]:NAME?

Parameter

name	Program name. Up to 32 ASCII characters without any control characters, space characters, single and double quotes, and comma. Parameter data type is SPD.

Query response

name <newline>

name returns the name of the program currently selected. Response data type is SRD.

Example

:PROG:NAME “program1”

:PROG:SEL:NAME?

:PROGram[:SELected]:STATe

Changes the execution status of a program stored in the program memory.

Before executing this command, the program must be selected by the :PROGram[:SELected]:NAME command. Or else, an error occurs.

Syntax

:PROGram[:SELected]:STATe *operation*

:PROGram[:SELected]:STATe?

Parameter

operation	RUN|PAUSe|STEP|STOP|CONTinue. Parameter data type is CPD. See Table 4-3 for the status changed by this command.

Query response

status <newline>

status returns the present execution status, Running, Paused, or Stopped. Response data type is CRD.

Example

:PROG:STAT PAUS

:PROG:SEL:STAT?

Table 4-3 Execution Status Changed by :PROG:STAT Command

operation	Present execution status		
	Running	**Paused**	**Stopped**
RUN	Error	to Running	to Running
PAUSe	to Paused	Paused	Stopped
STEP	Error	to Running to Paused	to Running to Paused
STOP	to Stopped	to Stopped	Stopped
CONTinue	Error	to Running	Error

:PROGram[:SELected]:WAIT?

Blocks other commands until the program execution status changes to Paused or Stopped.

Syntax :PROGram[:SELected]:WAIT? *timeout*

Parameter ***timeout*** Timeout value, in seconds. Parameter data type is NRf+.

Query response *status* <newline>

status returns 1 if the execution status changes to Paused or Stopped within the specified timeout, or 0 if a timeout occurs and the status is still in Running. Response data type is NR1.

Example :PROG:WAIT? 1

:PROGram:VARiable

Sets a value to the variable specified by *h*.

Variables can be used in the memory program. They must be expressed as %*h*% (*h*: integer. 1 to 100) in the memory program.

Syntax :PROGram:VARiable[*h*] *value*

:PROGram:VARiable[*h*]?

Parameter

value	Value of the variable specified by *h*. Up to 32 ASCII characters. Parameter data type is SPD.

Example

:PROG:VAR “1”

:PROG:VAR100?

READ Subsystem

:READ:ARRay?

Executes the :INITiate command and the :FETCh:ARRay? command in series, and returns the array data which contains all of the voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:READ:ARRay? [*chanlist*]

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the array data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr1,*ch1sour1*,*ch2curr1*,*ch2sour1*,*ch1curr2*,*ch1sour2*,*ch2curr2*,*ch2sour2*, ...
ch1curr5,*ch1sour5*,*ch2curr5*,*ch2sour5*,*ch1curr6*,*ch1sour6*,+9.910000E+37,+9.910000E+37, ...
ch1curr10,*ch1sour10*,+9.910000E+37,+9.910000E+37

This example shows the data containing the current data (*ch1currN*) and source data (*ch1sourN*) of the 10-step sweep measurement by channel 1, and the current data (*ch2currN*) and source data (*ch2sourN*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:READ:ARR? (@1,2)
```

:READ:ARRay:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>?

Executes the :INITiate command and the :FETCh:ARRay:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? command in series, and returns the array data which contains all of the current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:READ:ARRay:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? [*chanlist*]

For <CURRent|RESistance|SOURce|STATus|TIME|VOLTage>, specify CURRent for current measurement data, RESistance for resistance measurement data, SOURce for source output setting data, STATus for status data, TIME for time data, or VOLTage for voltage measurement data.

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the array data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr1,*ch2curr1*,*ch1curr2*,*ch2curr2*, ...
ch1curr5,*ch2curr5*,*ch1curr6*,+9.910000E+37, ... ,*ch1curr10*,+9.910000E+37

This example shows the data containing the current data (*ch1currN*) of the 10-step sweep measurement by channel 1, and the current data (*ch2currN*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example :READ:ARR:CURR? (@2,1)

:READ[:SCALar]?

Executes the :INITiate command and the :FETCh[:SCALar]? command in series, and returns the latest voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax :READ[:SCALar]? [*chanlist*]

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response *response* <newline>

response returns the latest data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr10,ch1sour10,ch2curr5,ch2sour5

This example shows the data containing the latest current data (*ch1curr10*) and source data (*ch1sour10*) of the 10-step sweep measurement by channel 1, and the latest current data (*ch2curr5*) and source data (*ch2sour5*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:READ? (@1,2)
```

:READ[:SCALar]:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>?

Executes the :INITiate command and the :FETCh[:SCALar]:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? command in series, and returns the latest current measurement data, resistance measurement data, source output setting data, status data, time data, or voltage measurement data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

:READ[:SCALar]:<CURRent|RESistance|SOURce|STATus|TIME|VOLTage>? [*chanlist*]

For <CURRent|RESistance|SOURce|STATus|TIME|VOLTage>, specify CURRent for current measurement data, RESistance for resistance measurement data, SOURce for source output setting data, STATus for status data, TIME for time data, or VOLTage for voltage measurement data.

Parameter

chanlist Channels to return the data. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the latest data specified by CURRent, RESistance, SOURce, STATus, TIME, or VOLTage. Response data type is NR3. See “Data Output Format” on page 1-12.

If both channels 1 and 2 are selected by *chanlist*, *response* returns the channel 1 data and the channel 2 data in this order. See the following example. With the ASCII data output format, each data is separated by a comma.

ch1curr10,ch2curr5

This example shows the data containing the latest current data (*ch1curr10*) of the 10-step sweep measurement by channel 1, and the latest current data (*ch2curr5*) of the 5-step sweep measurement by channel 2.

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example :READ:CURR? (@2,1)

SENSe Subsystem

For the numeric suffix [*c*], see “Numeric Suffix” on page 1-8.

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture

Sets the integration time for one point measurement.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture *time*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture? [*time*]

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, set CURRent[:DC], RESistance, or VOLTage[:DC]. Specifying the measurement item is not important because the *time* value is common for all items.

Parameter

time	*value* (+8E-6 to +2 seconds)|MINimum|MAXimum|DEFault (default is 0.1 PLC, =0.1/*power line frequency*). Parameter data type is NRf+. Query does not support *time=value*. If you specify the value less than MIN or greater than MAX, *time* is automatically set to MIN or MAX.

The integration time can be expressed by the following formula, using the NPLC value set by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles command. So the last command setting is effective for both *time* and *nplc*.

time = nplc / power line frequency

Query response

time <newline>

time returns the present setting of the integration time. If a parameter is specified, *time* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:CUR:APER 2E-3

:SENS2:CURR:DC:APER?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture:AUTO

Enables or disables the automatic aperture function.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture:AUTO *mode*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture:AUTO?

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, set CURRent[:DC], RESistance, or VOLTage[:DC]. Specifying the measurement item is not important because the *mode* value is common for all items.

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic aperture function.

mode=1 or ON enables the automatic aperture function. If this function is enabled, the instrument automatically sets the integration time (NPLC value) suitable for the measurement range.

The automatic aperture on/off works with the automatic NPLC on/off set by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles:AUTO command. So the last command setting is effective for both functions.

Query response

mode <newline>

mode is 0 or 1, and indicates that the automatic aperture function is off or on, respectively. Response data type is NR1.

Example

:SENS:CUR:APER:AUTO 0

:SENS2:CURR:DC:APER:AUTO?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles

Sets the number of power line cycles (NPLC) value instead of setting the integration time for one point measurement.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles *nplc*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles? [*nplc*]

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, set CURRent[:DC], RESistance, or VOLTage[:DC]. Specifying the measurement item is not important because the *nplc* value is common for all items.

Parameter

nplc *value* (+4E-4 to +100 for 50 Hz or +4.8E-4 to +120 for 60 Hz)|MINimum|MAXimum|DEFault (default is 0.1 PLC, power line cycles). Parameter data type is NRf+. Query does not support *nplc=value*. If you specify the value less than MIN or greater than MAX, *nplc* is automatically set to MIN or MAX.

The NPLC value can be expressed by the following formula, using the integration time set by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture command. So the last command setting is effective for both *nplc* and *time*.

nplc = time × power line frequency

Query response

nplc <newline>

nplc returns the present setting of the number of power line cycles. If a parameter is specified, *nplc* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:CUR:NPLC 0.2

:SENS2:CURR:DC:NPLC?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles:AUTO

Enables or disables the automatic NPLC function.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles:AUTO *mode*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:NPLCycles:AUTO?

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, set CURRent[:DC], RESistance, or VOLTage[:DC]. Specifying the measurement item is not important because the *mode* value is common for all items.

Parameter

mode 1|ON|0|OFF (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic NPLC function.

mode=1 or ON enables the automatic NPLC function. If this function is enabled, the instrument automatically sets the NPLC value (integration time) suitable for the measurement range.

The automatic NPLC on/off works with the automatic aperture on/off set by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:APERture:AUTO command. So the last command setting is effective for both functions.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the automatic NPLC function is off or on, respectively. Response data type is NR1.

Example :SENS:CUR:NPLC:AUTO 0

:SENS2:CURR:DC:NPLC:AUTO?

:SENSe:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]

Sets the compliance value of the specified channel.

Syntax :SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel] *compliance*

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]? [*compliance*]

For <CURRent[:DC]|VOLTage[:DC]>, specify CURRent[:DC] for current compliance, or VOLTage[:DC] for voltage compliance.

Parameter

compliance *value* (see maximum current or maximum voltage in "Source Output Ranges" on page 2-24) |MINimum| MAXimum|DEFault (default is 100 μA or 2 V). Parameter data type is NRf+. Effective values of *value* are from the minimum measurement value to the maximum measurement value of the channel. Query does not support *compliance=value*.

Query response *compliance* <newline>

compliance returns the present compliance value. If a parameter is specified, *compliance* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example :SENS:CURR:PROT 1E-3

:SENS2:CURR:DC:PROT:LEV?

:SENSe:<CURRent[:DC]|VOLTage[:DC]>:PROTection:TRIPped?

Returns the compliance status of the specified channel.

Syntax :SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection:TRIPped?

For <CURRent[:DC]|VOLTage[:DC]>, specify CURRent[:DC] for current compliance, or VOLTage[:DC] for voltage compliance.

Query response *status* <newline>

status is 1 or 0, and indicates if the channel is in the compliance state or not. Response data type is NR1.

Example :SENS:CURR:PROT:TRIP?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO

Enables or disables the automatic ranging function of the specified measurement channel.

Syntax :SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO *mode*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO?

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, specify CURRent[:DC] for current measurement, RESistance for resistance measurement, or VOLTage[:DC] for voltage measurement.

Parameter ***mode*** 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the automatic measurement ranging. If this function is disabled, the measurement is performed by using the range set by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe[:UPPer] command.

mode=1 or ON enables the automatic measurement ranging. If this function is enabled, the channel automatically sets the range which provides the best resolution to perform the measurement.

If a range is manually selected, automatic ranging is disabled.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the automatic measurement ranging is off or on, respectively. Response data type is NR1.

Example :SENS:CURR:RANG:AUTO 0

:SENS2:CURR:DC:RANG:AUTO?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO:LLIMit

Specifies the lower limit for the automatic measurement ranging operation, and sets the minimum measurement range which provides the best resolution to measure the specified value.

If the minimum measurement range is the same as the maximum measurement range, the measurement is performed by using this range.

An error occurs if the minimum range is greater than the maximum range.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO:LLIMit *range*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO:LLIMit? [*range*]

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, specify CURRent[:DC] for current measurement, RESistance for resistance measurement, or VOLTage[:DC] for voltage measurement.

Parameter

range *value*|MINimum|MAXimum|DEFault. Parameter data type is NRf+. Query does not support *range=value*.

value for current measurement: See Table 2-8.

value for voltage measurement: See Table 2-7.

value for resistance measurement: See Table 2-9. This is available for the resistance measurements set to the AUTO mode which is selected by the :SENSe:RESistance:MODE command.

Query response

range <newline>

range returns the present setting of the minimum measurement range for the auto range. If a parameter is specified, *range* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:SENS:CURR:RANG:AUTO:LLIM 1E-6
:SENS2:CURR:DC:RANG:AUTO:LLIM?
```

:SENSe:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:MODE

Selects the operation mode of the automatic measurement ranging. This command setting is not effective if the automatic ranging is disabled by the :SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO command.

Syntax

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:MODE *mode*

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:MODE?

For <CURRent[:DC]|VOLTage[:DC]>, specify CURRent[:DC] for current measurement, or VOLTage[:DC] for voltage measurement.

Parameter

mode NORMal (default)|RESolution|SPEed. Parameter data type is CPD.

mode=NORMal supports basic operation and downward changing operation described below

mode=RESolution supports basic operation and upward changing operation described below

mode=SPEed supports basic operation and upward and downward changing operations described below

- Basic operation

  The channel automatically sets the range which provides the best resolution to perform the measurement.

- Upward changing operation

  If measured data ≥ *value1*, the range shifts upward after a measurement.

  value1 = measurement range × *rate* / 100

- Downward changing operation

  If measured data ≤ *value2*, the range shifts downward immediately.

  value2 = measurement range × *rate* / 1000

rate value is set by the :SENSe:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:THReshold command.

Query response

mode <newline>

mode returns NORM, RES, or SPE, and indicates that the operation mode of automatic measurement ranging. Response data type is CRD.

Example

:SENS:CURR:RANG:AUTO:MODE SPE

:SENS2:CURR:DC:RANG:AUTO:MODE?

:SENSe:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:THReshold

Sets the threshold rate for the automatic measurement ranging operation.

Syntax

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:THReshold *rate*

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:RANGe:AUTO:THReshold? [*rate*]

For <CURRent[:DC]|VOLTage[:DC]>, specify CURRent[:DC] for current measurement, or VOLTage[:DC] for voltage measurement.

Parameter

rate *value*|MINimum|MAXimum|DEFault (default is 90 %). Parameter data type is NRf+. Effective values of *value* are 11 % to 100 %. Query does not support *rate=value*.

Query response

rate <newline>

rate returns the present setting of the threshold rate for automatic measurement ranging. If a parameter is specified, *rate* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:CURR:RANG:AUTO:THR 60

:SENS2:CURR:DC:RANG:AUTO:THR?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:AUTO:ULIMit

Specifies the upper limit for the automatic measurement ranging operation, and sets the maximum measurement range which provides the best resolution to measure the specified value. This is effective for resistance measurements set to the AUTO mode by the :SENSe:RESistance:MODE command.

For current measurement and voltage measurement, the maximum measurement range depends on the compliance range.

Syntax

:SENSe[*c*]:RESistance:RANGe:AUTO:ULIMit *range*

:SENSe[*c*]:RESistance:RANGe:AUTO:ULIMit? [*range*]

:SENSe[*c*]:<CURRent|VOLTage>[:DC]:RANGe:AUTO:ULIMit?

For <CURRent|VOLTage>, specify CURRent for the current measurement, or VOLTage for the voltage measurement.

Parameter

range *value* (see Table 2-9)|MINimum|MAXimum| DEFault (default is 200 MΩ). Parameter data type is NRf+. Query does not support *range=value*.

Query response

range <newline>

range returns the present setting of the maximum measurement range for the auto range. If a parameter is specified, *range* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:RES:RANG:AUTO:ULIM 1E6

:SENS2:CURR:DC:RANG:AUTO:ULIM?

:SENSe:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe[:UPPer]

Specifies the expected measurement value, and sets the measurement range which provides the best resolution to measure the specified value.

Syntax

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:UPPer *range*

:SENSe[*c*]:<CURRent[:DC]|RESistance|VOLTage[:DC]>:RANGe:UPPer? [*range*]

For <CURRent[:DC]|RESistance|VOLTage[:DC]>, specify CURRent[:DC] for current measurement, RESistance for resistance measurement, or VOLTage[:DC] for voltage measurement.

Parameter

range *value*|UP|DOWN|MINimum|MAXimum|DEFault. Parameter data type is NRf+. Query does not support *range=value*, UP, and DOWN.

value for current measurement: See Table 2-8.

value for voltage measurement: See Table 2-7.

value for resistance measurement: See Table 2-9. This is available for the resistance measurements set to the AUTO mode which is selected by the :SENSe:RESistance:MODE command.

range=UP sets the next higher measurement range.

range=DOWN sets the next lower measurement range.

Query response *range* <newline>

range returns the present setting of the measurement range. If a parameter is specified, *range* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example :SENS:CURR:RANG:UPP 1

:SENS2:CURR:DC:RANG:UPP?

:SENSe:DATA?

Returns the array data which contains all of the current measurement data, voltage measurement data, resistance measurement data, source output setting data, status data, or time data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax :SENSe[*c*]:DATA? [*offset*[, *size*]]

Parameter

offset Indicates the beginning of the data received. *n*|CURRent|STARt (default). Parameter data type is NR1 or CPD.

offset=*n* specifies the *n+1*th data. *n* is an integer, 0 to maximum (depends on the buffer state).

offset=CURR specifies the present data position.

offset=STAR specifies the top of the data buffer. Same as *offset*=0.

size Number of data to be received. 1 to maximum (depends on the buffer state). Parameter data type is NR1. If this parameter is not specified, all data from *offset* is returned.

Query response *response* <newline>

response returns the array data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

As shown in the following example, *response* may contain multiple data and elements. This example contains the current data (*ch1currN*) and source data (*ch1sourN*) of the 10-step sweep measurement by channel 1. With the ASCII data output format, each data is separated by a comma.

ch1curr1,ch1sour1,ch1curr2,ch1sour2, ... ch1curr10,ch1sour10

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:SENS:DATA?
```

:SENSe:DATA:LATest?

Returns the latest current measurement data, voltage measurement data, resistance measurement data, source output setting data, status data, or time data specified by the :FORMat:ELEMents:SENSe command. The data is not cleared until the :INITiate, :MEASure, or :READ command is executed.

Syntax

```
:SENSe:DATA:LATest?
```

Query response

response <newline>

response returns the latest data specified by the :FORMat:ELEMents:SENSe command. Response data type is NR3. See “Data Output Format” on page 1-12.

As shown in the following example, *response* may contain multiple data elements. This example contains the latest current data (*ch1curr10*) and source data (*ch1sour10*) of the 10-step sweep measurement by channel 1. With the ASCII data output format, each data is separated by a comma.

ch1curr10,ch1sour10

If the measurement function is not enabled or no data exists, *response* returns +9.910000E+37 (ASCII) or NaN (IEEE-754) which indicates “not a number”.

Example

```
:FORM:ELEM:SENS CURR,SOUR
:SENS:DATA:LAT?
```

:SENSe:FUNCtion:OFF

Disables the specified measurement functions.

Syntax

```
:SENSe[c]:FUNCtion:OFF function[, function[, function]]
```

:SENSe[*c*]:FUNCtion:OFF?

Parameter ***function*** "CURRent[:DC]"|"VOLTage[:DC]"|"RESistance" (default). Case insensitive. Parameter data type is SPD.

function="CURRent[:DC]" selects the current measurement function.

function="VOLTage[:DC]" selects the voltage measurement function.

function="RESistance" selects the resistance measurement function.

Query response *function*[, *function*[, *function*]]<newline>

function returns "CURR", "VOLT", or "RES", and indicates that the currently disabled measurement function. If a function is not selected, query returns "" (null string). Response data type is SRD.

Example :SENS:FUNC:OFF "RES","VOLT"

:SENS2:FUNC:OFF?

:SENSe:FUNCtion:OFF:ALL

Disables all measurement functions.

Syntax :SENSe[*c*]:FUNCtion:OFF:ALL

Example :SENS:FUNC:OFF:ALL

:SENSe:FUNCtion:OFF:COUNt?

Returns the number of measurement functions that are disabled.

Syntax :SENSe[*c*]:FUNCtion:OFF:COUNt?

Example :SENS:FUNC:OFF:COUN?

:SENSe:FUNCtion[:ON]

Enables the specified measurement functions.

Syntax :SENSe[*c*]:FUNCtion[:ON] *function*[, *function*[, *function*]]

:SENSe[*c*]:FUNCtion[:ON]?

Parameter ***function*** "CURRent[:DC]"|"VOLTage[:DC]"|"RESistance" Default is "VOLT","CURR". Case insensitive. Parameter data type is SPD.

function="CURRent[:DC]" selects the current measurement function.

function="VOLTage[:DC]" selects the voltage measurement function.

function="RESistance" selects the resistance measurement function. See ":SENSe:RESistance:MODE" on page 4-86 for resistance measurements.

Query response *function*[, *function*[, *function*]]<newline>

function returns "CURR", "VOLT", or "RES", and indicates that the currently enabled measurement function. If a function is not selected, query returns "" (null string). Response data type is SRD.

Example :SENS:FUNC "RES","VOLT"

:SENS2:FUNC:ON?

:SENSe:FUNCtion[:ON]:ALL

Enables all measurement functions.

See ":SENSe:RESistance:MODE" on page 4-86 for resistance measurements.

Syntax :SENSe[*c*]:FUNCtion[:ON]:ALL

Example :SENS:FUNC:ALL

:SENSe:FUNCtion[:ON]:COUNt?

Returns the number of measurement functions that are enabled.

Syntax :SENSe[*c*]:FUNCtion[:ON]:COUNt?

Example :SENS:FUNC:COUN?

:SENSe:FUNCtion:STATe?

Returns if the specified measurement function is enabled or disabled.

Syntax :SENSe[*c*]:FUNCtion:STATe? *function*

Parameter ***function*** "CURRent[:DC]"|"VOLTage[:DC]"|"RESistance". Parameter data type is SPD.

function="CURRent[:DC]" specifies the current measurement function.

function="VOLTage[:DC]" specifies the voltage measurement function.

function="RESistance" specifies the resistance measurement function.

Query response *response* <newline>

response returns 0 or 1, and indicates that the specified measurement function is now disabled or enabled respectively. Response data type is NR1.

Example :SENS:FUNC:STAT? "CURR"

:SENSe:REMote

Enables or disables the remote sensing. Remote sensing must be enabled to use the 4-wire connection (Kelvin connection).

Syntax :SENSe[*c*]:REMote *mode*

:SENSe[*c*]:REMote?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=0 or OFF disables remote sensing.

mode=1 or ON enables remote sensing.

Query response *mode* <newline>

mode is 0 or 1, and indicates that the remote sensing is off or on, respectively. Response data type is NR1.

Example :SENS:REM 1

:SENS2:REM?

:SENSe:RESistance:MODE

Selects the resistance measurement mode.

Syntax :SENSe[*c*]:RESistance:MODE *mode*

:SENSe[*c*]:RESistance:MODE?

Parameter ***mode*** MANual (default)|AUTO. Parameter data type is CPD.

If *mode*=MANual is selected, the source and measurement condition must be set manually. If the resistance measurement function is enabled, the resistance is calculated by *voltage*/*current*. The resistance measurement range cannot be set.

If *mode*=AUTO is selected, the channel automatically sets the current source and voltage measurement if the resistance measurement function is enabled. The current output value and range depend on the resistance measurement range selected by the :SENS:RES:RANG:UPP or SENS:RES:RANG:AUTO:LLIM and :SENS:RES:RANG:AUTO:ULIM commands. And the following parameters are set automatically.

Source output mode: CURRent

Source current automatic ranging: OFF

Source output shape: DC

Source current mode: FIXed

Voltage measurement range: 2 V range

Voltage compliance: 2.1 V

Voltage measurement automatic range: OFF

High capacitance mode: OFF

Query response *mode* <newline>

mode returns MAN or AUTO, and indicates the resistance measurement mode. Response data type is CRD.

Example :SENS:RES:MODE MAN

:SENS2:RES:MODE?

:SENSe:RESistance:OCOMpensated

Enables or disables the offset-compensated resistance measurement.

Syntax :SENSe[*c*]:RESistance:OCOMpensated *mode*

:SENSe[*c*]:RESistance:OCOMpensated?

Parameter ***mode*** 1|ON|0|OFF (default). Parameter data type is boolean.

mode=0 or OFF disables offset-compensated resistance measurement.

mode=1 or ON enables offset-compensated resistance measurement.

Query response *mode* <newline>

mode is 0 or 1, and indicates that offset-compensated resistance measurement is off or on, respectively. Response data type is NR1.

Example :SENS:RES:OCOM 1

:SENS2:RES:OCOM?

:SENSe:TOUTput:SIGNal

Selects the trigger output for the status change between the trigger layer and the acquire device action. Multiple trigger output ports can be set.

Syntax :SENSe[*c*]:TOUTput:SIGNal *output*{,*output*}

:SENSe[*c*]:TOUTput:SIGNal?

Parameter

output	Trigger output port. EXT1 (default)|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14|LAN|INT1|INT2. Parameter data type is CPD.

output=INT1 or INT2 selects the internal bus 1 or 2, respectively.

output=LAN selects a LAN port.

output=EXT*n* selects the GPIO pin *n*, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response *response* <newline>

response returns the present setting, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD. Multiple responses are separated by a comma.

Example :SENS:TOUT:SIGN EXT3

:SENS2:TOUT:SIGN?

:SENSe:TOUTput[:STATe]

Enables or disables the trigger output for the status change between the trigger layer and the acquire device action.

Syntax :SENSe[*c*]:TOUTput[:STATe] *mode*

:SENSe[*c*]:TOUTput[:STATe]?

Parameter

mode Trigger output ON or OFF. 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the trigger output.

mode=0 or OFF disables the trigger output.

Query response

response <newline>

response returns 1 or 0, and indicates that the trigger output is on or off, respectively. Response data type is NR1.

Example

:SENS:TOUT 1

:SENS2:TOUT:STAT?

:SENSe:WAIT:AUTO

Enables or disables the initial wait time used for calculating the measurement wait time for the specified channel. The initial wait time is automatically set by the instrument and cannot be changed. See :SENSe:WAIT[:STATe].

Syntax

:SENSe[*c*]:WAIT:AUTO *mode*

:SENSe[*c*]:WAIT:AUTO?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the initial wait time.

mode=0 or OFF disables the initial wait time. The initial wait time is set to 0.

Query response

mode <newline>

mode is 0 or 1, and indicates that the initial wait time is disabled or enabled, respectively. Response data type is NR1.

Example

:SENS:WAIT:AUTO 0

:SENS2:WAIT:AUTO?

:SENSe:WAIT:GAIN

Sets the gain value used for calculating the measurement wait time for the specified channel. See :SENSe:WAIT[:STATe].

Syntax

:SENSe[*c*]:WAIT:GAIN *gain*

:SENSe[*c*]:WAIT:GAIN? [*gain*]

Parameter

gain *value* (0 to 100)|MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf. Query does not support *gain=value*.

Query response

gain <newline>

gain returns the present setting of the gain value. If a parameter is specified, *gain* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:WAIT:GAIN 0.5

:SENS2:WAIT:GAIN?

:SENSe:WAIT:OFFSet

Sets the offset value used for calculating the measurement wait time for the specified channel. See :SENSe:WAIT[:STATe].

Syntax

:SENSe[*c*]:WAIT:OFFSet *offset*

:SENSe[*c*]:WAIT:OFFSet? [*offset*]

Parameter

offset *value* (0 to 1 seconds)|MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf. Query does not support *offset=value*.

Query response

offset <newline>

offset returns the present setting of the offset value. If a parameter is specified, *offset* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:SENS:WAIT:OFFS 0.5

:SENS2:WAIT:OFFS?

:SENSe:WAIT[:STATe]

Enables or disables the measurement wait time for the specified channel. The wait time is defined as the time the measurement channel cannot start measurement after the start of a DC output or the trailing edge of a pulse.

Syntax

:SENSe[*c*]:WAIT[:STATe] *mode*

:SENSe[*c*]:WAIT[:STATe]?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the measurement wait time. The wait time is set to 0.

mode=1 or ON enables the measurement wait time given by the following formula.

- :SENSe:WAIT:AUTO ON|1 condition:

  wait time = *gain* × initial wait time + *offset*

- :SENSe:WAIT:AUTO OFF|0 condition:

  wait time = *offset*

The initial wait time is automatically set by the instrument and cannot be changed.

gain and *offset* are set by the :SENSe:WAIT:GAIN and :SENSe:WAIT:OFFSet commands, respectively.

Query response

mode <newline>

mode is 0 or 1, and indicates that the measurement wait time is disabled or enabled, respectively. Response data type is NR1.

Example

:SENS:WAIT 0

:SENS2:WAIT:STAT?

SOURce Subsystem

For the numeric suffixes [*c*] and [*n*], see “Numeric Suffix” on page 1-8.

[:SOURce]:<CURRent|VOLTage>:<CENTer|SPAN>

Sets the center or span value of the current or voltage sweep output.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:<CENTer|SPAN> *data*

[:SOURce[*c*]]:<CURRent|VOLTage>:<CENTer|SPAN>? [*data*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

For <CENTer|SPAN>, specify CENTer for the sweep center value, or SPAN for the sweep span value.

Parameter

data	Sweep center or span value. *value* (see “Source Output Ranges” on page 2-24) |MINimum|MAXimum|DEFault (default is 0.0). Parameter data type is NRf+. Query does not support *data=value*.

The center and span values can be expressed by the following formula, using the start and stop values set by the [:SOURce]:<CURRent|VOLTage>:<STARt|STOP> command. So the last command setting is effective for these sweep parameters.

center = (*start* + *stop*)/2

span = *stop* - *start*

Query response

data <newline>

data returns the present setting. If a parameter is specified, *data* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:CURR:CENT 1E-3

:SOUR2:VOLT:SPAN?

[:SOURce]:<CURRent|VOLTage>[:LEVel][:IMMediate][:AMPLitude]

Changes the output level of the specified source channel immediately.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>[:LEVel][:IMMediate][:AMPLitude] *level*

[:SOURce[*c*]]:<CURRent|VOLTage>[:LEVel][:IMMediate][:AMPLitude]? [*level*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

level Current or voltage output level. *value* (see "Source Output Ranges" on page 2-24) |MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *level=value*.

Query response

level <newline>

level returns the present setting. If a parameter is specified, *level* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:VOLT 3

:SOUR2:CURR:LEV:IMM:AMPL?

[:SOURce]:<CURRent|VOLTage>[:LEVel]:TRIGgered[:AMPLitude]

Changes the output level of the specified source channel immediately by receiving a trigger source set by the :TRIGger<:ACQuire|:TRANsient|[:ALL]>:SOURce[:SIGNal] command.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>[:LEVel]:TRIGgered[:AMPLitude] *level*

[:SOURce[*c*]]:<CURRent|VOLTage>[:LEVel]:TRIGgered[:AMPLitude]? [*level*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

level Current or voltage output level. *value* (see "Source Output Ranges" on page 2-24) |MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *level=value*.

Query response

level <newline>

level returns the present setting. If a parameter is specified, *level* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:VOLT:TRIG 3

:SOUR2:CURR:LEV:TRIG:AMPL?

[:SOURce]:<CURRent|VOLTage>:MODE

Selects the source mode, fixed, list sweep, or sweep, of the specified source channel.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:MODE *mode*

[:SOURce[*c*]]:<CURRent|VOLTage>:MODE?

Parameter

mode Source mode. SWEep|LIST|FIXed (default). Parameter data type is CPD.

mode=FIX sets the constant current or voltage source.

mode=LIST sets the user-specified current or voltage list sweep source.

mode=SWEep sets the current or voltage sweep source.

Query response

mode <newline>

mode returns FIX, LIST, or SWE. Response data type is CRD.

Example

:VOLT:MODE SWE

:SOUR2:CURR:MODE?

[:SOURce]:<CURRent|VOLTage>:POINts

Sets the number of sweep steps for the current or voltage sweep output.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:POINts *points*

[:SOURce[*c*]]:<CURRent|VOLTage>:POINts? [*points*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

points Number of sweep steps. *value* (1 to 2500)|MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf+. Query does not support *points=value*.

The points value can be expressed by the following formula, using the step value set by the [:SOURce]:<CURRent|VOLTage>:STEP command and the span value set by the [:SOURce]:<CURRent|VOLTage>:<CENTer|SPAN> command.

points = *span*/*step* + 1 (where *step* is not 0)

points=1 sets *step*=0.

If *points* is changed, *span* works as a constant and *step* is changed. If *step* is changed, *span* works as a constant and *points* is changed. If *span* is changed, *points* works as a constant and *step* is changed.

The calculated points value is rounded down to an integer.

The sweep measurement is performed from the *start* value to the *stop* value given by the following formula, even if the specified stop value does not satisfy it.

stop = *start* + *step* × (*points* - 1)

For the logarithmic sweep, the *step* value is ignored and is not used for the calculation of sweep points.

Query response *points* <newline>

points returns the present setting. If a parameter is specified, *points* returns the value assigned to DEF, MIN, or MAX. Response data type is NR1.

Example

```
:CURR:POIN 51
:SOUR2:VOLT:POIN?
```

[:SOURce]:<CURRent|VOLTage>:RANGe

Sets the current or voltage output range of the specified source channel. This command is effective when the automatic ranging function is off.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe *range*

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe?

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

range *value* (see “Source Output Ranges” on page 2-24) |MINimum| MAXimum|DEFault. Parameter data type is NRf+.

value for current output: See Table 2-6.

value for voltage output: See Table 2-4.

Query response *range* <newline>

range returns the present setting. Response data type is NR3.

Example

:CURR:RANG 1E-6

:SOUR2:VOLT:RANG?

[:SOURce]:<CURRent|VOLTage>:RANGe:AUTO

Enables or disables the automatic ranging function for the specified source channel.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe:AUTO *mode*

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe:AUTO?

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables automatic ranging. If this function is disabled, the source output is performed by using the range set by the [:SOURce]:<CURRent|VOLTage>:RANGe command.

mode=1 or ON enables automatic ranging. If this function is enabled, the channel automatically sets the range which provides the best resolution to apply the source output.

If a range is manually selected, automatic ranging is disabled.

Query response

mode <newline>

mode is 0 or 1, and indicates that automatic ranging is off or on, respectively. Response data type is NR1.

Example

:CURR:RANG:AUTO 0

:SOUR2:VOLT:RANG:AUTO?

[:SOURce]:<CURRent|VOLTage>:RANGe:AUTO:LLIMit

Specifies the lower limit for the automatic output ranging operation, and sets the minimum range which provides the best resolution to apply the specified value.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe:AUTO:LLIMit *range*

[:SOURce[*c*]]:<CURRent|VOLTage>:RANGe:AUTO:LLIMit? [*range*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

range *value* (see "Source Output Ranges" on page 2-24) |MINimum|MAXimum|DEFault. Parameter data type is NRf+. Query does not support *range=value*.

value for current output: See Table 2-6.

value for voltage output: See Table 2-4.

Query response

range <newline>

range returns the present setting. If a parameter is specified, *range* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:CURR:RANG:AUTO:LLIM 1E-6

:SOUR2:VOLT:RANG:AUTO:LLIM?

[:SOURce]:<CURRent|VOLTage>:<STARt|STOP>

Sets the start or stop value for the current or voltage sweep output.

Syntax

[:SOURce[*c*]]:<CURRent|VOLTage>:<STARt|STOP> *data*

[:SOURce[*c*]]:<CURRent|VOLTage>:<STARt|STOP>? [*data*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

For <STARt|STOP>, specify STARt for the sweep start value, or STOP for the sweep stop value.

Parameter

data Sweep start or stop value. *value* (see "Source Output Ranges" on page 2-24) |MINimum|MAXimum|DEFault (default is 0.0). Parameter data type is NRf+. Query does not support *data=value*.

The start and stop values can be expressed by the following formula, using the center and span values set by the [:SOURce]:<CURRent|VOLTage>:<CENTer|SPAN> command. So the last command setting is effective for these sweep parameters.

start = *center* - *span*/2

stop = *center* + *span*/2

Query response *data* <newline>

data returns the present setting. If a parameter is specified, *data* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example :VOLT:STOP 10

:SOUR2:CURR:STAR?

[:SOURce]:<CURRent|VOLTage>:STEP

Sets the sweep step value of the current or voltage sweep output.

Syntax [:SOURce[*c*]]:<CURRent|VOLTage>:STEP *step*

[:SOURce[*c*]]:<CURRent|VOLTage>:STEP? [*step*]

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

step	Sweep step value. *value* (see "Source Output Ranges" on page 2-24) |MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *step=value*.

The step value can be expressed by the following formula, using the points value set by the [:SOURce]:<CURRent|VOLTage>:POINts command and the span value set by the [:SOURce]:<CURRent|VOLTage>:<CENTer|SPAN> command.

step = *span*/(*points* - 1) (where *points* is not 1)

points=1 sets *step*=0.

If *points* is changed, *span* works as a constant and *step* is changed. If *step* is changed, *span* works as a constant and *points* is changed. If *span* is changed, *points* works as a constant and *step* is changed.

The calculated points value is rounded down to an integer.

The sweep measurement is performed from the *start* value to the *stop* value given by the following formula, even if the specified stop value does not satisfy it.

stop = *start* + *step* × (*points* - 1)

For the logarithmic sweep, the *step* value is ignored and is not used for the calculation of sweep points.

Polarity of *step* and *span* must be the same. Different polarity causes an error.

Query response *step* <newline>

step returns the present setting. If a parameter is specified, *step* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:VOLT:STEP 0.5

:SOUR2:CURR:STEP?

[:SOURce]:DIGital:DATA

Sets the output data to the GPIO pins (digital control port) and read data from the GPIO pins.

Syntax

[:SOURce]:DIGital:DATA *data*

[:SOURce]:DIGital:DATA?

Parameter

data Output data. *value* (0 to 16383)(default is 0). Parameter data type is NR1.

Query response

data <newline>

data returns the data read from the GPIO pins. Response data type is NR1 or NDN selected by the :FORMat:DIGital command.

Example

:DIG:DATA 2900

:SOUR:DIG:DATA?

[:SOURce]:DIGital:EXTernal[*n*]:FUNCtion

Assigns the input/output function to the specified GPIO pin.

Syntax

[:SOURce]:DIGital:EXTernal[*n*][:FUNCtion] *function*

[:SOURce]:DIGital:EXTernal[*n*][:FUNCtion]?

Parameter

function Function. DINPut (default for the EXT1 to EXT13 pins)|DIO|HVOL (default for the EXT14 pin)|TINPut|TOUT. Parameter data type is CPD.

function=DINP assigns the digital input.

function=DIO assigns the digital I/O.

function=HVOL assigns the high voltage status output. Only for the EXT14 pin.

function=TINP assigns the trigger input.

function=TOUT assigns the trigger output.

Query response *function* <newline>

function returns DIO, DINP, TOUT, or TINP. Response data type is CRD.

Example :DIG:EXT TOUT

:SOUR:DIG:EXT14:FUNC?

[:SOURce]:DIGital:EXTernal[*n*]:POLarity

Sets the polarity of the input/output function for the specified GPIO pin. The input/output function is set by the [:SOURce]:DIGital:EXTernal[n]:FUNCtion command

Syntax [:SOURce]:DIGital:EXTernal[*n*]:POLarity *polarity*

[:SOURce]:DIGital:EXTernal[*n*]:POLarity?

Parameter

polarity	Polarity of the input/output function. NEG (default for the EXT1 to EXT13 pins)|POS (default for the EXT14 pin). Parameter data type is CPD.

polarity=POS sets positive polarity.

polarity=NEG sets negative polarity.

Query response *polarity* <newline>

polarity returns POS or NEG. Response data type is CRD.

Example :DIG:EXT:POL NEG

:SOUR:DIG:EXT14:POL?

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:POSition

Selects the output trigger timing for the specified GPIO pin.

Syntax [:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:POSition *position*

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:POSition?

Parameter ***position*** Output trigger timing. BEFore|AFTer|BOTH (default). Parameter data type is CPD.

type=BEFore enables trigger output at the beginning of arm, trigger, and device actions (transient or acquire).

type=AFTer enables trigger output at the end of arm, trigger, and device actions (transient or acquire).

type=BOTH enables trigger output at both beginning and end of arm, trigger, and device actions (transient or acquire).

Query response *response* <newline>

response returns the present setting of output trigger timing, BEF, AFT or BOTH. Response data type is CRD.

Example

```
:DIG:EXT:TOUT:POS BEF
:SOUR:DIG:EXT2:TOUT:POS?
```

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:WIDTh

Sets the pulse width of the output trigger for the specified GPIO pin.

Syntax [:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:WIDTh *width*

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput[:EDGE]:WIDTh? [*width*]

Parameter ***width*** Pulse width. *value* (1E-5 to 1E-2, in seconds)| MINimum|MAXimum|DEFault (default is 0.1 ms). Parameter data type is NRf+. Query does not support *width=value*.

Query response *width* <newline>

width returns the present setting. If a parameter is specified, *width* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:DIG:EXT:TOUT:WIDT 1E-5
:SOUR:DIG:EXT14:TOUT:WIDT?
```

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput:TYPE

Selects the output trigger type for the specified GPIO pin.

Syntax

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput:TYPE *type*

[:SOURce]:DIGital:EXTernal[*n*]:TOUTput:TYPE?

Parameter

type Trigger type. EDGE (default)|LEVel. Parameter data type is CPD.

type=EDGE selects the Edge trigger.

type=LEVel selects the Level trigger.

Query response

response <newline>

response returns the present setting of trigger type, EDGE or LEV. Response data type is CRD.

Example

:DIG:EXT:TOUT:TYPE LEV

:SOUR:DIG:EXT14:TOUT:TYPE?

[:SOURce]:DIGital:INTernal[*c*]:TOUTput[:EDGE]:POSition

Selects the output trigger timing for the specified channel.

Syntax

[:SOURce]:DIGital:INTernal[*c*]:TOUTput[:EDGE]:POSition *position*

[:SOURce]:DIGital:INTernal[*c*]:TOUTput[:EDGE]:POSition?

Parameter

position Output trigger timing. BEFore|AFTer|BOTH (default). Parameter data type is CPD.

type=BEFore enables trigger output at the beginning of arm, trigger, and device actions (transient or acquire).

type=AFTer enables trigger output at the end of arm, trigger, and device actions (transient or acquire).

type=BOTH enables trigger output at both beginning and end of arm, trigger, and device actions (transient or acquire).

Query response

response <newline>

response returns the present setting of output trigger timing, BEF, AFT or BOTH. Response data type is CRD.

Example

:DIG:INT:TOUT:POS BEF

:SOUR:DIG:INT2:TOUT:POS?

[:SOURce]:FUNCtion:MODE

Selects the source output mode of the specified channel.

Syntax

[:SOURce[*c*]]:FUNCtion:MODE *mode*

[:SOURce[*c*]]:FUNCtion:MODE?

Parameter

mode Source output mode. CURRent|VOLTage (default). Parameter data type is CPD.

mode=CURR sets the specified channel to the current source. Voltage compliance for the current source is set by the :SENS:VOLT:PROT[:LEV] command.

mode=VOLT sets the specified channel to the voltage source. Current compliance for the voltage source is set by the :SENS:CURR:PROT[:LEV] command.

See ":SENSe:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]" on page 4-76.

Query response

mode <newline>

mode returns CURR or VOLT. Response data type is CRD.

Example

:FUNC:MODE CURR

:SOUR2:FUNC:MODE?

[:SOURce]:FUNCtion[:SHAPe]

Selects the source output shape of the specified channel.

Syntax

[:SOURce[*c*]]:FUNCtion[:SHAPe] *shape*

[:SOURce[*c*]]:FUNCtion[:SHAPe]?

Parameter

shape Source output shape. PULSe|DC (default). Parameter data type is CPD.

shape=DC sets the specified channel to DC (constant) output.

shape=PULS sets the specified channel to pulsed output.

Query response

shape <newline>

shape returns DC or PULS. Response data type is CRD.

Example

:FUNC PULS

:SOUR2:FUNC:SHAP?

[:SOURce]:FUNCtion:TRIGgered:CONTinuous

Enables or disables continuous trigger output for the specified channel.

Syntax

[:SOURce[*c*]]:FUNCtion:TRIGgered:CONTinuous *mode*

[:SOURce[*c*]]:FUNCtion:TRIGgered:CONTinuous?

Parameter

mode 0|OFF (default)|1|ON. Parameter data type is boolean.

mode=1 or ON enables continuous trigger output. The specified channel keeps the output level and range settings even after the grouped channels change status from busy to idle. The last output settings are saved as the immediate output settings.

mode=0 or OFF disables continuous trigger output. The specified channel changes the output level and range settings to the previous settings immediately when the grouped channels change status from busy to idle. The previous settings must be set by the [:SOURce]:<CURRent|VOLTage>[:LEVel][:IMMediate][:AMPLitude] command and the range setup command.

Query response

mode <newline>

mode returns 0 or 1, and indicates that continuous trigger is off or on, respectively. Response data type is NR1.

Example

:FUNC:TRIG:CONT 0

:SOUR2:FUNC:TRIG:CONT?

[:SOURce]:LIST:<CURRent|VOLTage>

Sets the source output current or voltage data for the specified channel.

Syntax

[:SOURce[*c*]]:LIST:<CURRent|VOLTage> *list*

[:SOURce[*c*]]:LIST:<CURRent|VOLTage>?

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter ***list*** List of the output current or voltage data. Default is 0. Parameter data type is NRf.

Maximum of 2500 data can be set to *list*. Each data must be separated by a comma, for example: *list=0.1,0.2,0.3*. For effective values of the output current or voltage data, see "Source Output Ranges" on page 2-24.

Query response *list* <newline>

list returns the present setting of the list. Multiple data is separated by a comma. Response data type is NR3.

Example :LIST:VOLT 0.1,0.2,0.3

:SOUR2:LIST:CURR?

[:SOURce]:LIST:<CURRent|VOLTage>:APPend

Adds the source output current or voltage data to the end of the list set by the [:SOURce]:LIST:<CURRent|VOLTage> command, to which some data might be appended to by this command. Total number of data in the list must be ≤ 2500.

Syntax [:SOURce[*c*]]:LIST:<CURRent|VOLTage>:APPend *append_list*

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter ***append_list*** List of the output current or voltage data. Parameter data type is NRf+.

Multiple data can be set to *append_list*. Each data must be separated by a comma, for example: *append_list=1.1,1.2,1.3*. For effective values of the output current or voltage data, see "Source Output Ranges" on page 2-24.

Example :LIST:VOLT:APP 1.1,1.2,1.3

:SOUR2:LIST:CURR:APP 1E-6,2E-6,3E-6

[:SOURce]:LIST:<CURRent|VOLTage>:POINts?

Returns the number of data in the list set by the [:SOURce]:LIST:<CURRent|VOLTage> command, to which some data might be appended to by the [:SOURce]:LIST:<CURRent|VOLTage>:APPend command.

Syntax [:SOURce[*c*]]:LIST:<CURRent|VOLTage>:POINts?

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Query response *number_of_data* <newline>

number_of_data returns the number of data in the list. Response data type is NR1.

Example :LIST:VOLT:POIN?

:SOUR2:LIST:CURR:POIN?

[:SOURce]:LIST:<CURRent|VOLTage>:STARt

Specifies the list sweep start point by using the index of the list.

Syntax [:SOURce[*c*]]:LIST:<CURRent|VOLTage>:STARt *start*

[:SOURce[*c*]]:LIST:<CURRent|VOLTage>:STARt?

For <CURRent|VOLTage>, specify CURRent for current output, or VOLTage for voltage output.

Parameter

start	Index of the list. 1 to 2500. Default is 1. Parameter data type is NR1. *start*=1 indicates the first data in the list (top of the list). *start*=0 or the value greater than 2500 causes an error.

Query response *start* <newline>

start returns the present setting of the list sweep start point. Response data type is NR1.

Example :LIST:VOLT:STAR 10

:SOUR2:LIST:CURR:STAR?

[:SOURce]:PULSe:DELay

Sets the pulse delay time for the specified channel. The pulse delay time is the time from starting the pulse base output to starting the pulse level transition (or to starting the pulse peak output).

Syntax [:SOURce[*c*]]:PULSe:DELay *delay*

[:SOURce[*c*]]:PULSe:DELay? [*delay*]

Parameter ***delay*** Delay time. *value* (0.0 to 99999.9, in seconds)|MINimum| MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *delay=value*.

Query response *delay* <newline>

delay returns the present setting. If a parameter is specified, *delay* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:PULS:DEL 1E-3
:SOUR2:PULS:DEL?
```

[:SOURce]:PULSe:WIDTh

Sets the pulse width for the specified channel. The pulse width is the time from starting the pulse peak output (or starting the pulse level transition) to the end of the pulse peak output. However, it is strictly defined as the time from 10 % of peak level at the leading edge to 90 % of peak level at the trailing edge.

Syntax

[:SOURce[*c*]]:PULSe:WIDTh *width*

[:SOURce[*c*]]:PULSe:WIDTh? [*width*]

Parameter ***width*** Pulse width. *value* (5E-5 to 100000 seconds, in 1E-6 resolution)|MINimum| MAXimum|DEFault (default is 5E-5). Parameter data type is NRf+. Query does not support *width=value*.

Minimum time for the pulse base output is also 50 µs. And the minimum pulse period is 100 µs.

Query response *width* <newline>

width returns the present setting. If a parameter is specified, *width* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:PULS:WIDT 2E-2
:SOUR2:PULS:WIDT?
```

[:SOURce]:SWEep:DIRection

Sets the sweep direction, UP or DOWN, for the specified channel.

Syntax

[:SOURce[*c*]]:SWEep:DIRection *direction*

[:SOURce[*c*]]:SWEep:DIRection?

Parameter

direction Sweep direction. DOWN|UP (default). Parameter data type is CPD.

direction=UP sets the sweep direction from start value to stop value. The sweep measurement is performed from the *start* value to the *stop* value given by the following formula, even if the specified stop value does not satisfy it.

$stop = start + step \times (points - 1)$

direction=DOWN sets the sweep direction from stop value to start value. The sweep measurement is performed from the *stop* value to the *start* value given by the following formula, even if the specified start value does not satisfy it.

$start = stop - step \times (points - 1)$

Query response

direction <newline>

direction returns the present setting of the sweep direction, UP or DOWN. Response data type is CRD.

Example

:SWE:DIR DOWN

:SOUR2:SWE:DIR?

[:SOURce]:SWEep:POINts

Sets the number of sweep steps for the specified channel. This command setting is effective for both current sweep and voltage sweep.

Syntax

[:SOURce[*c*]]:SWEep:POINts *points*

[:SOURce[*c*]]:SWEep:POINts? MINimum| MAXimum|DEFault

Parameter

points Number of sweep steps. *value* (1 to 2500)|MINimum| MAXimum|DEFault (default is 1). Parameter data type is NRf+.

The points value can be expressed by the following formula, using the step value set by the [:SOURce]:<CURRent|VOLTage>:STEP command and the span value set by the [:SOURce]:<CURRent|VOLTage>:<CENTer|SPAN> command.

$points = span/step + 1$ (where *step* is not 0)

points=1 sets *step*=0.

If *points* is changed, *span* works as a constant and *step* is changed. If *step* is changed, *span* works as a constant and *points* is changed. If *span* is changed, *points* works as a constant and *step* is changed.

The calculated points value is rounded down to an integer.

The sweep measurement is performed from the *start* value to the *stop* value given by the following formula, even if the specified stop value does not satisfy it.

$stop = start + step \times (points - 1)$

For the logarithmic sweep, the *step* value is ignored and is not used for the calculation of sweep points.

Query response *points* <newline>

points returns the value assigned to DEF, MIN, or MAX. Response data type is NR1.

Example :SWE:POIN 51

:SOUR2:SWE:POIN? MAX

[:SOURce]:SWEep:RANGing

Selects the output ranging mode of the sweep output for the specified channel.

Syntax [:SOURce[*c*]]:SWEep:RANGing *mode*

[:SOURce[*c*]]:SWEep:RANGing?

Parameter

mode	Ranging mode. BEST (default)|FIXed|AUTO. Parameter data type is CPD.

If *mode*=BEST is set, the channel automatically sets the range which covers the whole sweep output level for the linear sweep (SPACing *mode*=LINear), or the range which provides the best resolution to apply the source output for each step of the log sweep (SPACing *mode*=LOGarithmic).

If *mode*=AUTO is set, the channel automatically sets the range which provides the best resolution to apply the source output for each sweep step.

If *mode*=FIX is set, the channel uses only the range effective when starting the sweep. Range change is not performed while the sweep output is applied.

Query response *mode* <newline>

mode returns the present setting of the output ranging mode, BEST, FIX, or AUTO. Response data type is CRD.

Example

:SWE:RANG BEST

:SOUR2:SWE:RANG?

[:SOURce]:SWEep:SPACing

Selects the scale of the sweep output for the specified channel.

Syntax

[:SOURce[*c*]]:SWEep:SPACing *mode*

[:SOURce[*c*]]:SWEep:SPACing?

Parameter

mode	Sweep scale. LOGarithmic|LINear (default). Parameter data type is CPD.

mode=LIN selects the linear scale sweep output.

mode=LOG selects the logarithmic scale sweep output. For the log sweep, the sweep step value is ignored.

Query response

mode <newline>

mode returns the present setting of the scale, LIN or LOG. Response data type is CRD.

Example

:SWE:SPAC LOG

:SOUR2:SWE:SPAC?

[:SOURce]:SWEep:STAir

Sets the sweep mode for the specified channel.

Syntax

[:SOURce[*c*]]:SWEep:STAir *mode*

[:SOURce[*c*]]:SWEep:STAir?

Parameter

mode	Sweep mode. SINGle (default)|DOUBle. Parameter data type is CPD.

mode=SINGle sets the sweep mode to single sweep.

mode=DOUBle sets the sweep mode to double sweep. Double sweep performs the sweep from start to stop to start.

Query response *mode* <newline>

mode returns SING or DOUB, and indicates that the sweep mode is single or double, respectively. Response data type is CRD.

Example :SWE:STA DOUB

:SOUR2:SWE:STA?

[:SOURce]:TOUTput:SIGNal

Selects the trigger output for the status change between the trigger layer and the transient device action. Multiple trigger output ports can be set.

Syntax [:SOURce[*c*]]:TOUTput:SIGNal *output*{,*output*}

[:SOURce[*c*]]:TOUTput:SIGNal?

Parameter

output	Trigger output port. EXT1 (default)|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14|LAN|INT1|INT2. Parameter data type is CPD.

output=INT1 or INT2 selects the internal bus 1 or 2, respectively.

output=LAN selects a LAN port.

output=EXT*n* selects the GPIO pin *n*, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response *response* <newline>

response returns the present setting, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD. Multiple responses are separated by a comma.

Example :TOUT:SIGN EXT3

:SOUR2:TOUT:SIGN?

[:SOURce]:TOUTput[:STATe]

Enables or disables the trigger output for the status change between the trigger layer and the transient device action.

Syntax

[:SOURce[*c*]]:TOUTput[:STATe] *mode*

[:SOURce[*c*]]:TOUTput[:STATe]?

Parameter

mode Trigger output ON or OFF. 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the trigger output.

mode=0 or OFF disables the trigger output.

Query response

response <newline>

response returns 1 or 0, and indicates that the trigger output is on or off, respectively. Response data type is NR1.

Example

:TOUT 1

:SOUR2:TOUT:STAT?

[:SOURce]:WAIT:AUTO

Enables or disables the initial wait time used for calculating the source wait time for the specified channel. The initial wait time is automatically set by the instrument and cannot be changed. See [:SOURce]:WAIT[:STATe].

Syntax

[:SOURce[*c*]]:WAIT:AUTO *mode*

[:SOURce[*c*]]:WAIT:AUTO?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the initial wait time.

mode=0 or OFF disables the initial wait time. The initial wait time is set to 0.

Query response

mode <newline>

mode is 0 or 1, and indicates that the initial wait time is disabled or enabled, respectively. Response data type is NR1.

Example

:WAIT:AUTO 0

:SOUR2:WAIT:AUTO?

[:SOURce]:WAIT:GAIN

Sets the gain value used for calculating the source wait time for the specified channel. See [:SOURce]:WAIT[:STATe].

Syntax

[:SOURce[*c*]]:WAIT:GAIN *gain*

[:SOURce[*c*]]:WAIT:GAIN? [*gain*]

Parameter

gain *value* (0 to 100)|MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf. Query does not support *gain=value*.

Query response

gain <newline>

gain returns the present setting of the gain value. If a parameter is specified, *gain* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:WAIT:GAIN 0.5

:SOUR2:WAIT:GAIN?

[:SOURce]:WAIT:OFFSet

Sets the offset value used for calculating the source wait time for the specified channel. See [:SOURce]:WAIT[:STATe].

Syntax

[:SOURce[*c*]]:WAIT:OFFSet *offset*

[:SOURce[*c*]]:WAIT:OFFSet? [*offset*]

Parameter

offset *value* (0 to 1 seconds)|MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf. Query does not support *offset=value*.

Query response

offset <newline>

offset returns the present setting of the offset value. If a parameter is specified, *offset* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:WAIT:OFFS 0.5

:SOUR2:WAIT:OFFS?

[:SOURce]:WAIT[:STATe]

Enables or disables the source wait time for the specified channel. This wait time is defined as the time the source channel cannot change the output after the start of a DC output or the trailing edge of a pulse.

Syntax

[:SOURce[*c*]]:WAIT[:STATe] *mode*

[:SOURce[*c*]]:WAIT[:STATe]?

Parameter

mode 0|OFF|1|ON (default). Parameter data type is boolean.

mode=0 or OFF disables the source wait time. The wait time is set to 0.

mode=1 or ON enables the source wait time given by the following formula.

- [:SOURce]:WAIT:AUTO ON|1 condition:

  wait time = *gain* × initial wait time + *offset*

- [:SOURce]:WAIT:AUTO OFF|0 condition:

  wait time = *offset*

The initial wait time is automatically set by the instrument and cannot be changed.

gain and *offset* are set by the [:SOURce]:WAIT:GAIN and [:SOURce]:WAIT:OFFSet commands respectively.

Query response

mode <newline>

mode is 0 or 1, and indicates that the source wait time is disabled or enabled, respectively. Response data type is NR1.

Example

:WAIT 0

STATus Subsystem

:STATus:<MEASurement|OPERation|QUEStionable>:CONDition?

Returns the value of the measurement, operation, or questionable status condition register. See Table 4-4 to 4-6 for the bit definitions. The register setting is not changed by this command.

Syntax

:STATus:<MEASurement|OPERation|QUEStionable>:CONDition?

For <MEASurement|OPERation|QUEStionable>, specify MEASurement for the measurement status condition register, OPERation for the operation status condition register, or QUEStionable for the questionable status condition register.

Query response

value <newline>

value returns the value of the specified register. It is the sum of the binary-weighted values for the set bits. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:MEAS:COND?

:STAT:OPER:COND?

:STAT:QUES:COND?

Table 4-4 Questionable Status Condition Register Bit Definitions

bit	decimal value	description	definition
0	1	Voltage Summary	Over voltage in channel 1, and/or 2.
1	2	Current Summary	Over current in channel 1, and/or 2.
2	4	Ch1 Output Protection	Output relay of the specified channel is opened by the automatic output off at compliance function.
3	8	Ch2 Output Protection	
4	16	Temperature Summary	Over temperature in channel 1, and/or 2.
5 to 7		Not used	0 is returned.

bit	decimal value	description	definition
8	256	Calibration	Channel 1 and/or 2 failed calibration.
9	512	Self-test	Channel 1 and/or 2 failed self-test.
10	1024	Interlock	Interlock circuit is open.
11	2048	Ch1 Transition Event Lost	Lost arm or trigger transition event on channel 1
12	4096	Ch1 Acquire Event Lost	Lost arm or trigger acquire event on channel 1
13	8192	Ch2 Transition Event Lost	Lost arm or trigger transition event on channel 2
14	16384	Ch2 Acquire Event Lost	Lost arm or trigger acquire event on channel 2
15		Not used	0 is returned.

Table 4-5 Measurement Status Condition Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Limit Test Summary	Channel 1 failed one or more limit tests.
1	2	Ch1 Reading Available	Reading of channel 1 was taken normally.
2	4	Ch1 Reading Overflow	Reading of channel 1 exceeds the selected measurement range.
3	8	Ch1 Buffer Available	Trace buffer for channel 1 has data.
4	16	Ch1 Buffer Full	Trace buffer for channel 1 is full.
5	32	Not used	0 is returned.
6	64	Ch2 Limit Test Summary	Channel 2 failed one or more limit tests.
7	128	Ch2 Reading Available	Reading of channel 2 was taken normally.
8	256	Ch2 Reading Overflow	Reading of channel 2 exceeds the selected measurement range.
9	512	Ch2 Buffer Available	Trace buffer for channel 2 has data.
10	1024	Ch2 Buffer Full	Trace buffer for channel 2 is full.
11 to 15		Not used	0 is returned.

Table 4-6 Operation Status Condition Register Bit Definitions

bit	decimal value	description	definition
0	1	Calibration/Self-test Running	Self-calibration or Self-test is in progress.
1	2	Ch1 Transition Idle	Channel 1 is in the transition idle state.
2	4	Ch1 Waiting for Transition Trigger	Channel 1 is waiting for the transition trigger.
3	8	Ch1 Waiting for Transition Arm	Channel 1 is waiting for the transition arm.
4	16	Ch1 Acquire Idle	Channel 1 is in the acquire idle state.
5	32	Ch1 Waiting for Acquire Trigger	Channel 1 is waiting for the acquire trigger.
6	64	Ch1 Waiting for Acquire Arm	Channel 1 is waiting for the acquire arm.
7	128	Ch2 Transition Idle	Channel 2 is in the transition idle state.
8	256	Ch2 Waiting for Transition Trigger	Channel 2 is waiting for the transition trigger.
9	512	Ch2 Waiting for Transition Arm	Channel 2 is waiting for the transition arm.
10	1024	Ch2 Acquire Idle	Channel 2 is in the acquire idle state.
11	2048	Ch2 Waiting for Acquire Trigger	Channel 2 is waiting for the acquire trigger.
12	4056	Ch2 Waiting for Acquire Arm	Channel 2 is waiting for the acquire arm.
13	8192	Instrument Locked	If a remote interface (GPIB, USB, or LAN) has a lock (see :SYSTem:LOCK:OWNer? command), this bit will be set. When a remote interface releases the lock (see :SYSTem:LOCK:NAME? command), this bit will be cleared.
14	16384	Program Running	Program is running. 0 is set during the program memory execution is stopped.
15	32768	Not used	0 is returned.

:STATus:<MEASurement|OPERation|QUEStionable>:ENABle

Sets the measurement, operation, or questionable status enable register. The enable register is a mask which allows true conditions in the event register to be reported in the summary bit.

Syntax

:STATus:<MEASurement|OPERation|QUEStionable>:ENABle *mask*

:STATus:<MEASurement|OPERation|QUEStionable>:ENABle?

For <MEASurement|OPERation|QUEStionable>, specify MEASurement for the measurement status enable register, OPERation for the operation status enable register, or QUEStionable for the questionable status enable register.

Parameter

mask	Mask. 0 to 65535 (decimal). Default is 0. Parameter data type is NR1 or NDN.

mask is the sum of the binary-weighted values for the set bits.

Query response

mask <newline>

mask returns the present setting of the specified enable register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:MEAS:ENAB 65535

:STAT:QUES:ENAB?

:STATus:<MEASurement|OPERation|QUEStionable>[:EVENt]?

Returns the value of the measurement, operation, or questionable status event register. The register setting is changed by this command.

Syntax

:STATus:<MEASurement|OPERation|QUEStionable>[:EVENt]?

For <MEASurement|OPERation|QUEStionable>, specify MEASurement for the measurement status event register, OPERation for the operation status event register, or QUEStionable for the questionable status event register.

Query response

value <newline>

value returns the present setting of the specified event register. It is the sum of the binary-weighted values for the set bits. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:MEAS:EVEN?

:STAT:OPER:EVEN?

:STAT:QUES:EVEN?

:STATus:<MEASurement|OPERation|QUEStionable>:NTRansition

Sets the negative transition filter in the measurement, operation, or questionable status register. If you set a bit of the filter, a 1-to-0 transition of its register bit sets the corresponding bit of the event register.

Syntax

:STATus:<MEASurement|OPERation|QUEStionable>:NTRansition *filter*

:STATus:<MEASurement|OPERation|QUEStionable>:NTRansition?

For <MEASurement|OPERation|QUEStionable>, specify MEASurement for the measurement status register, OPERation for the operation status register, or QUEStionable for the questionable status register.

Parameter

filter Negative transition filter. 0 to 65535 (decimal). Default is 0. Parameter data type is NR1 or NDN.

filter is the sum of the binary-weighted values for the set bits.

Query response

filter <newline>

filter returns the present setting of the negative transition filter in the specified register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:MEAS:NTR 0

:STAT:QUES:NTR?

:STATus:<MEASurement|OPERation|QUEStionable>:PTRansition

Sets the positive transition filter in the measurement, operation, or questionable status register. If you set a bit of the filter, a 0-to-1 transition of its register bit sets the corresponding bit of the event register.

Syntax

:STATus:<MEASurement|OPERation|QUEStionable>:PTRansition *filter*

:STATus:<MEASurement|OPERation|QUEStionable>:PTRansition?

For <MEASurement|OPERation|QUEStionable>, specify MEASurement for the measurement status register, OPERation for the operation status register, or QUEStionable for the questionable status register.

Parameter

filter	Positive transition filter. 0 to 65535 (decimal). Default is 32767. Parameter data type is NR1 or NDN.

filter is the sum of the binary-weighted values for the set bits.

Query response

filter <newline>

filter returns the present setting of the positive transition filter in the specified register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

```
:STAT:MEAS:PTR 32767
:STAT:QUES:PTR?
```

:STATus:PRESet

Sets all defined bits in the status system's PTR registers and clears the all bits in the NTR and Enable registers. The registers are returned to the default condition.

Syntax

:STATus:PRESet

Example

```
:STAT:PRES
```

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:CONDition?

Returns the value of the questionable status condition register. See Table 4-7 to 4-11 for the bit definitions. The register setting is not changed by this command.

Syntax

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:CONDition?

For <CALibration|CURRent|TEMPerature|TEST|VOLTage>, specify CALibration for the questionable calibration status condition register, CURRent for the questionable current status condition register, TEMPerature for the questionable temperature status condition register, TEST for the questionable self-test status condition register, or VOLTage for the questionable voltage status condition register.

Query response

value <newline>

value returns the value of the specified register. It is the sum of the binary-weighted values for the set bits. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:QUES:CAL:COND?

:STAT:QUES:CURR:COND?

:STAT:QUES:TEMP:COND?

:STAT:QUES:TEST:COND?

:STAT:QUES:VOLT:COND?

Table 4-7 Questionable Calibration Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Calibration Failed	Calibration fail in channel 1.
1	2	Ch2 Calibration Failed	Calibration fail in channel 2.
3 to 15		Not used	0 is returned.

Table 4-8 Questionable Current Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Over Current	Over current in channel 1.
1	2	Ch2 Over Current	Over current in channel 2.
3 to 15		Not used	0 is returned.

Table 4-9 Questionable Temperature Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Over Temperature	Over temperature in channel 1.
1	2	Ch2 Over Temperature	Over temperature in channel 2.
3 to 15		Not used	0 is returned.

Table 4-10 Questionable Test Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Self-test Failed	Self-test failure in channel 1.
1	2	Ch2 Self-test Failed	Self-test failure in channel 2.
3 to 15		Not used	0 is returned.

Table 4-11 Questionable Voltage Register Bit Definitions

bit	decimal value	description	definition
0	1	Ch1 Over Voltage	Over voltage in channel 1.
1	2	Ch2 Over Voltage	Over voltage in channel 2.
3 to 15		Not used	0 is returned.

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:ENABle

Sets the questionable calibration, current, temperature, test, or voltage status enable register. The enable register is a mask which allows true conditions in the event register to be reported in the summary bit.

Syntax

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:ENABle *mask*

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:ENABle?

For <CALibration|CURRent|TEMPerature|TEST|VOLTage>, specify CALibration for the questionable calibration status condition register, CURRent for the questionable current status condition register, TEMPerature for the questionable temperature status condition register, TEST for the questionable self-test status condition register, or VOLTage for the questionable voltage status condition register.

Parameter

mask Mask. 0 to 65535 (decimal). Default is 0. Parameter data type is NR1 or NDN.

mask is the sum of the binary-weighted values for the set bits.

Query response

mask <newline>

mask returns the present setting of the specified enable register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:QUES:CURR:ENAB 65535

:STAT:QUES:TEMP:ENAB?

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>[:EVENt]?

Returns the value of the questionable calibration, current, temperature, test, or voltage status event register. The register setting is changed by this command.

Syntax

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>[:EVENt]?

For <CALibration|CURRent|TEMPerature|TEST|VOLTage>, specify CALibration for the questionable calibration status condition register, CURRent for the questionable current status condition register, TEMPerature for the questionable temperature status condition register, TEST for the questionable self-test status condition register, or VOLTage for the questionable voltage status condition register.

Query response *value* <newline>

value returns the present setting of the specified event register. It is the sum of the binary-weighted values for the set bits. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example :STAT:QUES:CURR:EVEN?

:STAT:QUES:VOLT:EVEN?

:STAT:QUES:TEMP:EVEN?

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:NTRansition

Sets the negative transition filter in the questionable calibration, current, temperature, test, or voltage status register. If you set a bit of the filter, a 1-to-0 transition of its register bit sets the corresponding bit of the event register.

Syntax :STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:NTRansition *filter*

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:NTRansition?

For <CALibration|CURRent|TEMPerature|TEST|VOLTage>, specify CALibration for the questionable calibration status condition register, CURRent for the questionable current status condition register, TEMPerature for the questionable temperature status condition register, TEST for the questionable self-test status condition register, or VOLTage for the questionable voltage status condition register.

Parameter

filter	Negative transition filter. 0 to 65535 (decimal). Default is 0. Parameter data type is NR1 or NDN.

filter is the sum of the binary-weighted values for the set bits.

Query response *filter* <newline>

filter returns the present setting of the negative transition filter in the specified register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:QUES:CURR:NTR 0

:STAT:QUES:TEMP:NTR?

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:PTRansition

Sets the positive transition filter in the questionable calibration, current, temperature, test, or voltage status register. If you set a bit of the filter, a 0-to-1 transition of its register bit sets the corresponding bit of the event register.

Syntax

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:PTRansition *filter*

:STATus:QUEStionable:<CALibration|CURRent|TEMPerature|TEST|VOLTage>:PTRansition?

For <CALibration|CURRent|TEMPerature|TEST|VOLTage>, specify CALibration for the questionable calibration status condition register, CURRent for the questionable current status condition register, TEMPerature for the questionable temperature status condition register, TEST for the questionable self-test status condition register, or VOLTage for the questionable voltage status condition register.

Parameter

filter	Positive transition filter. 0 to 65535 (decimal). Default is 32767. Parameter data type is NR1 or NDN.

filter is the sum of the binary-weighted values for the set bits.

Query response

filter <newline>

filter returns the present setting of the positive transition filter in the specified register. Response data type is NR1 (decimal) or NDN (binary, octal, or hexadecimal) selected by the :FORMat:SREGister command.

Example

:STAT:QUES:CURR:PTR 32767

:STAT:QUES:TEMP:PTR?

SYSTem Subsystem

For the numeric suffix [*c*], see “Numeric Suffix” on page 1-8.

:SYSTem:BEEPer[:IMMediate]

Generates a beep sound of the specified frequency and duration.

Syntax :SYSTem:BEEPer[:IMMediate] *frequency*, *time*

Parameter

frequency	Frequency, in Hz. 55 to 6640 Hz. Parameter data type is NRf.
time	Duration, in seconds. 0.05 to 12.75 seconds. Parameter data type is NRf+.

Example :SYST:BEEP 100,0.5

:SYSTem:BEEPer:STATe

Enables or disables the beeper. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:BEEPer:STATe *mode*

:SYSTem:BEEPer:STATe?

Parameter

mode	Beeper on or off. 0|OFF|1|ON. Parameter data type is boolean.

mode=1 or ON enables the beeper.

mode=0 or OFF disables the beeper.

Query response *mode* <newline>

mode returns 0 or 1, and indicates that the beeper is off or on, respectively. Response data type is NR1.

Example :SYST:BEEP:STAT 1

:SYST:BEEP:STAT?

:SYSTem:COMMunicate:ENABle

Enables or disables the remote interface GPIB, USB, or LAN, the remote service Sockets, Telnet, VXI-11, HiSLIP, or the built-in Web Interface. The setting is effective after rebooting the instrument. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:COMMunicate:ENABle *mode*, *interface*

:SYSTem:COMMunicate:ENABle? *interface*

Parameter

mode Interface on or off. 1|ON|0|OFF. Parameter data type is boolean.

interface Interface. GPIB|USB|LAN|SOCKets|TELNet|VXI11|HISLip|WEB. Parameter data type is CPD.

mode=1 or ON enables the specified *interface*.

mode=0 or OFF disables the specified *interface*.

Query response

mode <newline>

mode returns 0 or 1, and indicates that the specified *interface* is off or on, respectively. Response data type is NR1.

Example

:SYST:COMM:ENAB 0,USB

:SYST:COMM:ENAB? LAN

:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess

Sets the GPIB address of the instrument. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess *address*

:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?

Parameter

address GPIB address, 0 to 30. Parameter data type is NR1.

Query response

address <newline>

address returns the GPIB address of the instrument. Response data type is NR1.

Example

:SYST:COMM:GPIB:ADDR 17

:SYST:COMM:GPIB:ADDR?

:SYSTem:COMMunicate:LAN:ADDRess

Sets the static LAN (IP) address of the instrument. The setting is enabled by the :SYSTem:COMMunicate:LAN:UPDate command. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:COMMunicate:LAN:ADDRess *address*

:SYSTem:COMMunicate:LAN:ADDRess? [CURRent|STATic]

Parameter

address IP address of the instrument. It must be in the *A.B.C.D* format with 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 225. Parameter data type is SPD.

Query response

address <newline>

address returns the static LAN (IP) address of the instrument. If the CURRent parameter is set, *address* returns the present setting. If the STATic parameter is set, *address* returns the reserved value for the next startup. Response data type is SRD.

Example

:SYST:COMM:LAN:ADDR “192.168.100.100”

:SYST:COMM:LAN:ADDR?

:SYSTem:COMMunicate:LAN:BSTatus?

Returns the LAN boot status of the instrument.

Syntax

:SYSTem:COMMunicate:LAN:BSTatus?

Query response

status <newline>

status returns the following LAN boot status. Response data type is CRD.

LAN_AUTO_IP The instrument booted with a local IP address.

LAN_DHCP The instrument booted with a DHCP-assigned address.

LAN_FAULT The instrument cannot detect a connection.

LAN_STATIC The instrument booted with a static IP address.

Example

:SYST:COMM:LAN:BST?

:SYSTem:COMMunicate:<LAN|TCPip>:CONTrol?

Returns the control connection port number of the specified port.

Syntax :SYSTem:COMMunicate:<LAN|TCPip>:CONTrol?

Query response *port_number* <newline>

port_number returns the control connection port number of the specified port. Response data type is NR1.

Example :SYST:COMM:TCP:CONT?

:SYSTem:COMMunicate:LAN:DHCP

Enables or disables the use of the Dynamic Host Configuration Protocol (DHCP). The setting is enabled by the :SYSTem:COMMunicate:LAN:UPDate command. This command setting is not changed by power off or the *RST command.

When DHCP is enabled, the instrument will try to obtain an IP address from a DHCP server. If a DHCP server finds the instrument, it will assign a dynamic IP address, subnet mask, and default gateway to the instrument. When DHCP is disabled or unavailable, the instrument will use the static IP address, subnet mask, and default gateway during power-on.

If a DHCP LAN address is not assigned by a DHCP server, a static IP address will be used after a timeout of approximately 2 minutes. For the instrument boot status, see the :SYSTem:COMMunicate:LAN:BSTatus? command.

Syntax :SYSTem:COMMunicate:LAN:DHCP *mode*

:SYSTem:COMMunicate:LAN:DHCP?

Parameter ***mode*** DHCP off or on. 0|OFF|1|ON. Parameter data type is boolean.

Query response *mode* <newline>

mode returns 0 or 1, and indicates that DHCP is off or on, respectively. Response data type is NR1.

Example :SYST:COMM:LAN:DHCP 0

:SYST:COMM:LAN:DHCP?

:SYSTem:COMMunicate:LAN:DNS

Sets the IP address of the DNS server. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:COMMunicate:LAN:DNS[*c*] *address*

:SYSTem:COMMunicate:LAN:DNS[*c*]? [CURRent|STATic]

Parameter

address	IP address of the DNS server. It must be in the *A.B.C.D* format with 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. Parameter data type is SPD.

Query response

address <newline>

address returns the IP address of the DNS server. If the CURRent parameter is set, *address* returns the present setting. If the STATic parameter is set, *address* returns the reserved value for the next startup. Response data type is SRD.

Example

:SYST:COMM:LAN:DNS “192.168.100.200”

:SYST:COMM:LAN:DNS2?

:SYSTem:COMMunicate:LAN:DOMain?

Returns the domain name of the network to which the instrument is connected.

Syntax

:SYSTem:COMMunicate:LAN:DOMain?

Query response

domain_name <newline>

domain_name returns the domain name of the network. Response data type is SRD.

Example

:SYST:COMM:LAN:DOM?

:SYSTem:COMMunicate:LAN:<GATE|GATeway>

Sets the IP address of the default gateway. The setting is enabled by the :SYSTem:COMMunicate:LAN:UPDate command. This command setting is not changed by power off or the *RST command. For <GATE|GATeway>, specify GATE or GATeway.

Syntax

:SYSTem:COMMunicate:LAN:<GATE|GATeway> *address*

:SYSTem:COMMunicate:LAN:<GATE|GATeway>? [CURRent|STATic]

Parameter ***address*** IP address of the default gateway. It must be in the *A.B.C.D* format with 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 225. Parameter data type is SPD.

Query response *address* <newline>

address returns the IP address of the default gateway. If the CURRent parameter is set, *address* returns the present setting. If the STATic parameter is set, *address* returns the reserved value for the next startup. Response data type is SRD.

Example :SYST:COMM:LAN:GATE “192.168.100.210”

:SYST:COMM:LAN:GATE?

:SYSTem:COMMunicate:LAN:<HNAMe|HOSTname>

Sets the host name of the instrument. The setting is enabled by the :SYSTem:COMMunicate:LAN:UPDate command. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:COMMunicate:LAN:<HNAMe|HOSTname> *hostname*

:SYSTem:COMMunicate:LAN:<HNAMe|HOSTname>? [CURRent|STATic]

Parameter ***hostname*** Host name. Up to 15 characters. Parameter data type is SPD.

Query response *hostname* <newline>

hostname returns the host name of the instrument. If the CURRent parameter is set, *hostname* returns the present setting. If the STATic parameter is set, *hostname* returns the reserved value for the next startup. Response data type is SRD.

Example :SYST:COMM:LAN:HNAM “A-B2911A-00001”

:SYST:COMM:LAN:HOST?

:SYSTem:COMMunicate:LAN:MAC?

Returns the MAC address of the instrument.

Syntax :SYSTem:COMMunicate:LAN:MAC?

Query response *mac_address* <newline>

mac_address returns the MAC address of the instrument. Response data type is SRD.

Example :SYST:COMM:LAN:MAC?

:SYSTem:COMMunicate:LAN:SMASk

Sets the static subnet mask. The setting is enabled by the :SYSTem:COMMunicate:LAN:UPDate command. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:COMMunicate:LAN:SMASk *subnet_mask*

:SYSTem:COMMunicate:LAN:SMASk? [CURRent|STATic]

Parameter

subnet_mask Subnet mask. It must be in the *A.B.C.D* format with 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. Parameter data type is SPD.

Query response *subnet_mask* <newline>

subnet_mask returns the subnet mask. If the CURRent parameter is set, *subnet_mask* returns the present setting. If the STATic parameter is set, *subnet_mask* returns the reserved value for the next startup. Response data type is SRD.

Example :SYST:COMM:LAN:SMAS “255.255.255.0”

:SYST:COMM:LAN:SMAS?

:SYSTem:COMMunicate:LAN:TELNet:PROMpt

Sets the command prompt displayed during a Telnet session for establishing communication with the instrument. This command setting is not changed by power off or the *RST command.

The instrument uses LAN port 5024 for SCPI Telnet sessions, and 5025 for SCPI Socket sessions.

A Telnet session can typically be started as shown below from a host computer shell.

telnet *ip_address port*

Syntax :SYSTem:COMMunicate:LAN:TELNet:PROMpt *prompt*

:SYSTem:COMMunicate:LAN:TELNet:PROMpt?

Parameter ***prompt*** Command prompt. Up to 15 characters. Parameter data type is SPD.

Query response *prompt* <newline>

prompt returns the command prompt. Response data type is SRD.

Example :SYST:COMM:LAN:TELN:PROM “A-B2911A-00001> ”

:SYST:COMM:LAN:TELN:PROM?

:SYSTem:COMMunicate:LAN:TELNet:WMESsage

Sets the welcome message displayed during a Telnet session when starting communication with the instrument. This command setting is not changed by power off or the *RST command.

The instrument uses LAN port 5024 for SCPI Telnet sessions, and 5025 for SCPI Socket sessions.

Syntax :SYSTem:COMMunicate:LAN:TELNet:WMESsage *message*

:SYSTem:COMMunicate:LAN:TELNet:WMESsage?

Parameter ***message*** Welcome message. Up to 63 characters. Parameter data type is SPD.

Query response *message* <newline>

message returns the welcome message. Response data type is SRD.

Example :SYST:COMM:LAN:TELN:WMES “Welcome to A-B2911A-00001.”

:SYST:COMM:LAN:TELN:WMES?

:SYSTem:COMMunicate:LAN:UPDate

Disconnects all active LAN and Web Interface connections, updates the LAN setup, and restarts the LAN interface with the new setup. The new setup may change the IP address of the instrument.

Syntax :SYSTem:COMMunicate:LAN:UPDate

Example :SYST:COMM:LAN:UPD

:SYSTem:COMMunicate:LAN:WINS

Sets the IP address of the WINS server. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:COMMunicate:LAN:WINS[*c*] *address*

:SYSTem:COMMunicate:LAN:WINS[*c*]? [CURRent|STATic]

Parameter

address	IP address of the WINS server. It must be in the *A.B.C.D* format with 15 characters maximum. *A*, *B*, *C*, and *D* must be a number from 0 to 255. Parameter data type is SPD.

Query response

address <newline>

address returns the IP address of the WINS server. If the CURRent parameter is set, *address* returns the present setting. If the STATic parameter is set, *address* returns the reserved value for the next startup. Response data type is SRD.

Example

:SYST:COMM:LAN:WINS “192.168.100.150”

:SYST:COMM:LAN:WINS2?

:SYSTem:DATA:QUANtity?

Returns the number of data for the specified channel in the data buffer.

Syntax

:SYSTem:DATA:QUANtity? [*chanlist*]

Parameter

chanlist	Channels. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Query response

response <newline>

response returns the number of data. Response data type is NR1. If both channels 1 and 2 are selected by *chanlist*, *response* returns the number of channel 1 data and the number of channel 2 data in this order. They are separated by a comma.

Example

:SYST:DATA:QUAN? (@2)

:SYSTem:DATE

Sets the date of the internal clock. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:DATE *year*, *month*, *day*

:SYSTem:DATE?

Parameter

year	Year. 4-digit integer. Parameter data type is NR1.
month	Month. Integer from 1 to 12. Parameter data type is NR1.
day	Day. Integer from 1 to 31. Parameter data type is NR1.

Query response

response <newline>

response returns *year*, *month*, *day*. Each value is separated by a comma. Response data type is NR1.

Example

:SYST:DATE 2011,1,1

:SYSTem:ERRor:ALL?

Reads and returns all items in the error/event queue, and clears the queue.

Syntax

:SYSTem:ERRor:ALL?

Query response

response <newline>

response returns *code*,*message* which contains the error/event code and message. Multiple responses are listed in the FIFO (first-in-first-out) order, separated by a comma. Data type of *code* is NR1 and *message* is SRD.

If the queue is empty, the response is +0,“No error”.

Example

:SYST:ERR:ALL?

:SYSTem:ERRor:CODE:ALL?

Reads all items in the error/event queue, returns all codes, and clears the queue.

Syntax

:SYSTem:ERRor:CODE:ALL?

Query response

code <newline>

code returns the error/event code. Multiple responses are listed in the FIFO (first-in-first-out) order, separated by a comma. Response data type is NR1.

If the queue is empty, the response is +0.

Example :SYST:ERR:CODE:ALL?

:SYSTem:ERRor:CODE[:NEXT]?

Reads and removes the top item in the error/event queue, and returns the top code.

Syntax :SYSTem:ERRor:CODE[:NEXT]?

Query response *code* <newline>

code returns the error/event code. Response data type is NR1.

If the queue is empty, the response is +0.

Example :SYST:ERR:CODE?

:SYSTem:ERRor:COUNt?

Returns the number of items in the error/event queue.

Syntax :SYSTem:ERRor:COUNt?

Query response *response* <newline>

response returns the number of items. Response data type is NR1.

If the queue is empty, the response is +0.

Example :SYST:ERR:COUN?

:SYSTem:ERRor[:NEXT]?

Reads and removes the top item in the error/event queue, and returns the top code and message.

Syntax :SYSTem:ERRor[:NEXT]?

Query response *response* <newline>

response returns *code,message* which contains the error/event code and message. Multiple responses are listed in the FIFO (first-in-first-out) order, separated by a comma. Data type of *code* is NR1 and *message* is SRD.

If the queue is empty, the response is +0,“No error”.

Example :SYST:ERR?

:SYSTem:FAN:MODE

Sets the fan control mode. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:FAN:MODE *mode*

:SYSTem:FAN:MODE?

Parameter ***mode*** Fan control mode, NORMal|RACK. Parameter data type is CPD.

mode=NORM is for normal use.

mode=RACK is for using in rack

Query response *mode* <newline>

mode returns the present setting, NORM or RACK. Response data type is CRD.

Example :SYST:FAN:MODE RACK

:SYST:FAN:MODE?

:SYSTem:GROup[:DEFine]

Defines the channel group. This command setting is not changed by power off or the *RST command.

The channel grouping is used to control the channel output timing automatically so that the channel keeps the output while the other channel performs measurement. The grouped channels start the source output in the order of the channel number, then start the measurement at the same time, and keep the output until the measurement is completed.

If the grouping is released, the channels work independently regardless of the condition of the other channel.

Syntax :SYSTem:GROup[:DEFine] *grouplist*

:SYSTem:GROup[:DEFine]?

Parameter ***grouplist*** Channel group setting. Parameter data type is channel list. See "Channel List Parameter" on page 1-8.

grouplist=(@1,2) for making the group of the channels 1 and 2, or
grouplist=(@1),(@2) for breaking the group.

Query response *grouplist* <newline>

grouplist returns the channel group setting. Response data type is channel list.

Example :SYST:GRO (@1,2)

:SYST:GRO:DEF?

:SYSTem:GROup:RESet

Releases the channel group defined by the :SYSTem:GROup[:DEFine] command.

Syntax :SYSTem:GROup:RESet

Example :SYST:GRO:RES

:SYSTem:INTerlock:TRIPped?

Returns if the interlock circuit is close or open.

Syntax :SYSTem:INTerlock:TRIPped?

Query response *mode* <newline>

mode returns 0 or 1, and indicates that the interlock circuit is close or open, respectively. Response data type is NR1.

Example :SYST:INT:TRIP?

:SYSTem:LANGuage

Selects the B2900 control command set. If the setting is changed, the instrument will be automatically rebooted. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:LANGuage *mode*

:SYSTem:LANGuage?

Parameter

mode B2900 control command set. “DEFault”|“2400”. Parameter data type is SPD.

mode=“DEF” selects the default command set which supports all B2900 functions.

mode=“2400” selects the conventional command set designed for existing programs which you created for controlling existing instruments, such as Series 2400 by Keithley Instruments, Inc. Once this mode is set, B2900 does not support the SCPI commands described in this manual, but only the SCPI commands listed in “Conventional commands supported by B2900” on page 6-3. For using this mode, see Chapter 6, “Using Your Existing Programs.”

Query response

mode <newline>

mode returns the present setting, DEF or 2400. Response data type is SRD.

Example

:SYST:LANG “2400”

:SYST:LANG?

:SYSTem:LFRequency

Selects the line frequency. This command setting is not changed by power off or the *RST command.

Syntax

:SYSTem:LFRequency *frequency*

:SYSTem:LFRequency?

Parameter

frequency Line frequency. 50 (for 50 Hz)|60 (for 60 Hz). Parameter data type is NR1.

Query response

frequency <newline>

frequency returns the present setting, 50 or 60. Response data type is NR1.

Example

:SYST:LFR 60

:SYST:LFR?

:SYSTem:LOCK:NAME?

Returns the current I/O interface (the I/O interface in use by the querying computer).

Syntax :SYSTem:LOCK:NAME?

Query response *response* <newline>

response returns GPIB, USB, VXI11, or LAN <IP Address>, indicating the I/O interface being used by the querying computer.

Example :SYST:LOCK:NAME?

Remarks Use this command to determine the interface you are currently using. Then use the :SYSTem:LOCK:OWNer? command to determine which interface, if any, has the lock.

:SYSTem:LOCK:OWNer?

Returns the I/O interface that currently has a lock.

Syntax :SYSTem:LOCK:OWNer?

Query response *response* <newline>

response returns GPIB, USB, VXI11, or LAN <IP Address>, which indicates the I/O interface. If no interface has a lock, then NONE is returned.

Example :SYST:LOCK:OWN?

Remarks When a lock is active, Bit 13 in the Standard Operation Register will be set (see :STATus:<MEASurement|OPERation|QUEStionable>:CONDition? command). When the lock is released on all I/O interfaces, this bit will be cleared.

:SYSTem:LOCK:RELease

Decrements the lock count by one, and may release the I/O interface from which the command is executed.

Syntax :SYSTem:LOCK:RELease

Example :SYST:LOCK:REL

Remarks

When a lock is active, Bit 13 in the Standard Operation Register will be set (see :STATus:<MEASurement|OPERation|QUEStionable>:CONDition? command). When the lock is released on all I/O interfaces, this bit will be cleared.

Note that for each successful lock request, a lock release is required. Two requests require two releases.

:SYSTem:LOCK:REQuest?

Requests a lock of the current I/O interface. This provides a mechanism by which you can lock the instrument's configuration or cooperatively share the instrument with other computers.

Syntax

:SYSTem:LOCK:REQuest?

Query response

response <newline>

response returns 1 if the lock request is granted, or 0 if denied.

Example

:SYST:LOCK:REQ?

Remarks

Lock requests can be nested, and each request increases the lock count by 1. For each request, you will need to issue a release from the same I/O interface (see :SYSTem:LOCK:RELease command).

Instrument locks are handled at the I/O interface level (GPIB, USB, LAN, etc.), and you are responsible for all coordination between threads and/or programs on that interface.

When a request is granted, only I/O sessions from the present interface will be allowed to change the state of the instrument. From other I/O interfaces, you can query the state of the instrument, but no measurement configuration changes or measurements will be allowed.

Locks from LAN sessions will be automatically released when a LAN disconnect is detected.

When a lock is granted, Bit 13 in the Standard Operation Register will be set (see :STATus:<MEASurement|OPERation|QUEStionable>:CONDition? command). In addition, the entire instrument front panel, including the Local key, will be locked down while a lock is in place (“KEYBOARD LOCKED” is displayed).

:SYSTem:PON

Specifies the power-on state.

The power-on state can be selected from the factory default reset condition (RST) and user conditions RCL0, RCL1, RCL2, RCL3, and RCL4 which can be defined by the *SAV 0, *SAV 1, *SAV 2, *SAV 3, and *SAV 4 commands, respectively.

Syntax :SYSTtem:PON *memory*

Parameter ***memory*** Power-on state, RST(default)|RCL0|RCL1|RCL2|RCL3|RCL4
Parameter data type is CPD.

Example :SYST:PON RCL0

:SYSTem:PRESet

Presets the instrument settings and the front panel display.

Syntax :SYSTem:PRESet

Example :SYST:PRESet

:SYSTem:SET

Sends or loads the instrument setup data.

Syntax :SYSTem:SET *data*

:SYSTem:SET?

Parameter ***data*** Instrument setup data. Parameter data type is a definite length arbitrary binary block.

Query response Response is a definite length arbitrary binary block.

:SYSTem:TIME

Sets the time of the internal clock. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:TIME *hour*, *minute*, *second*

:SYSTem:TIME?

Parameter ***hour*** Hour. Integer from 0 to 23. Parameter data type is NR1.

minute Minute. Integer from 0 to 59. Parameter data type is NR1.

second Second. Integer from 0 to 59. Parameter data type is NR1.

Query response *response* <newline>

response returns *hour*, *minute*, *second*. Each value is separated by a comma. Response data type is NR1.

Example :SYST:TIME 23,59,59

:SYSTem:TIME:TIMer:COUNt?

Returns the present count of the timer.

Syntax :SYSTem:TIME:TIMer:COUNt?

Query response *response* <newline>

response returns the present timer count. Response data type is NR3.

Example :SYST:TIME:TIM:COUN?

:SYSTem:TIME:TIMer:COUNt:RESet:AUTO

Enables or disables the automatic reset function of the timer. If this function is enabled, the timer count is reset when the initiate action occurs.

Syntax :SYSTem:TIME:TIMer:COUNt:RESet:AUTO *mode*

:SYSTem:TIME:TIMer:COUNt:RESet:AUTO?

Parameter ***mode*** Automatic reset function on or off. 0|OFF|1|ON (default). Parameter data type is boolean.

mode=1 or ON enables the automatic reset function.

mode=0 or OFF disables the automatic reset function.

Query response *mode* <newline>

mode returns 0 or 1, and indicates that the automatic reset function is off or on, respectively. Response data type is NR1.

Example :SYST:TIME:TIM:COUN:RES:AUTO 0

:SYST:TIME:TIM:COUN:RES:AUTO?

:SYSTem:TIME:TIMer:COUNt:RESet[:IMMediate]

Resets the timer count immediately.

Syntax :SYSTem:TIME:TIMer:COUNt:RESet[:IMMediate]

Example :SYST:TIME:TIM:COUN:RES

:SYSTem:VERSion?

Returns the version of the SCPI standard. This command setting is not changed by power off or the *RST command.

Syntax :SYSTem:VERSion?

Query response *response* <newline>

response returns the version of the SCPI standard. For example, *1999.0*. Response data type is NR2.

Example :SYST:VERS?

TRACe Subsystem

For the numeric suffix [*c*], see "Numeric Suffix" on page 1-8.

:TRACe:CLEar

Clears the trace buffer of the specified channel. This command is effective when the trace buffer control mode is set to NEV by the :TRACe:FEED:CONTrol command.

Syntax :TRACe[*c*]:CLEar

Example :TRAC2:CLE

:TRACe:DATA?

Returns data in the trace buffer. The data placed in the buffer is specified by the :TRACe:FEED command.

Syntax :TRACe[*c*]:DATA? [*offset*[, *size*]]

Parameter

offset Indicates the beginning of the data received. *n*|CURRent|STARt (default). Parameter data type is NR1 or CPD.

offset=*n* specifies the *n+1*th data. *n* is an integer, 0 to maximum (depends on the buffer state).

offset=CURR specifies the present data position.

offset=STAR specifies the top of trace buffer. Same as *offset*=0.

size Number of data to be received. 1 to maximum (depends on the buffer state). Parameter data type is NR1. If this parameter is not specified, all data from *offset* is returned.

Query response *data* <newline>

Response data type is NR3. See "Data Output Format" on page 1-12.

Example :TRAC2:DATA? 0,10

:TRACe:FEED

Specifies the data placed in the trace buffer. This command is effective when the trace buffer control mode is set to NEV by the :TRACe:FEED:CONTrol command.

Syntax

:TRACe[*c*]:FEED *type*

:TRACe[*c*]:FEED?

Parameter

type Data type. MATH|LIMit|SENSe (default). Parameter data type is CPD.

type=SENS specifies the measurement result data, which contains all of the voltage measurement data, current measurement data, resistance measurement data, time data, status data, or source output setting data specified by the :FORMat:ELEMents:SENSe command.

type=LIM specifies the limit test data. The data contains the limit test data, time data, or status data specified by the :FORMat:ELEMents:CALCulate command. See :CALCulate:DATA? for the limit test data.

type=MATH specifies the calculation result data. The data contains the calculation result, time data, or status data specified by the :FORMat:ELEMents:CALCulate command. See :CALCulate:MATH:DATA? for more information.

Query response

type <newline>

type returns the present setting of data type, MATH, LIM, or SENS. Response data type is CRD.

Example

:TRAC:FEED MATH

:TRAC2:FEED?

:TRACe:FEED:CONTrol

Selects the trace buffer control.

Syntax

:TRACe[*c*]:FEED:CONTrol *mode*

:TRACe[*c*]:FEED:CONTrol?

Parameter

mode Trace buffer control mode. NEXT|NEVer (default). Parameter data type is CPD.

mode=NEV disables write operation to the trace buffer. The :TRACe:CLEar, :TRACe:FEED, and :TRACe:POINts commands can be used.

mode=NEXT enables write operation until buffer full. Buffer full changes *mode* to NEV automatically. No error occurs.

Query response *mode* <newline>

mode returns the present setting of the control mode, NEXT or NEV. Response data type is CRD.

Example :TRAC:FEED:CONT NEXT

:TRAC2:FEED:CONT?

:TRACe:FREE?

Returns the available size (*available*) and the total size (*total*) of the trace buffer.

Syntax :TRACe[*c*]:FREE?

Query response *response* <newline>

response returns *available*,*total*. Each value is separated by a comma. Response data type is NR1.

Example :TRAC2:FREE?

:TRACe:POINts

Sets the size of the trace buffer. This command is effective when the trace buffer control mode is set to NEV by the :TRACe:FEED:CONTrol command.

Syntax :TRACe[*c*]:POINts *points*

:TRACe[*c*]:POINts? [*points*]

Parameter

points	Size. *value* (1 to 100000)|MINimum|MAXimum|DEFault (default is 100000). Parameter data type is NR1. Query does not support *points=value*.

Query response *points* <newline>

points returns the present setting of the buffer size. If a parameter is specified, *points* returns the value assigned to DEF, MIN, or MAX. Response data type is NR1.

Example :TRAC:POIN 10000

:TRAC2:POIN?

:TRACe:POINts:ACTual?

Returns the number of data in the trace buffer.

Syntax :TRACe[*c*]:POINts:ACTual?

Query response *points* <newline>

points returns the number of data in the trace buffer. Response data type is NR1.

Example :TRAC2:POIN:ACT?

:TRACe:STATistic:DATA?

Returns the result of the statistical operation for the data stored in the trace buffer. Before executing this command, the statistical operation must be specified by the :TRACe:STATistic:FORMat command.

If the trace buffer is storing raw measurement data for multiple data types, the statistical operation is performed for all measurement data.

Statistical operation is not available for the TIME and STATus data.

Syntax :TRACe[*c*]:STATistic:DATA?

Query response *response* <newline>

response returns the result of the statistical operation. Response data type is NR3. See “Data Output Format” on page 1-12.

Example :TRAC:STAT:DATA?

:TRACe:STATistic:FORMat

Selects the statistical operation performed by the :TRACe:STATistic:DATA? command.

Syntax :TRACe[*c*]:STATistic:FORMat *operation*

:TRACe[*c*]:STATistic:FORMat?

Parameter ***operation*** Statistical operation. MINimum|MAXimum|SDEViation|PKPK|MEAN (default). Parameter data type is CPD.

operation=MEAN sets the operation for obtaining the mean value.

operation=SDEV sets the operation for obtaining the standard deviation.

operation=PKPK sets the operation for obtaining the peak to peak value.

operation=MIN sets the operation for obtaining the minimum value.

operation=MAX sets the operation for obtaining the maximum value.

Query response *operation* <newline>

operation returns the present setting of the statistical operation, MEAN, SDEV, PKPK, MIN, or MAX. Response data type is CRD.

Example :TRAC:STAT:FORM PKPK

:TRAC2:STAT:FORM?

:TRACe:TSTamp:FORMat

Selects the rule for reading the timestamp data in the trace buffer.

Syntax :TRACe[*c*]:TSTamp:FORMat *rule*

:TRACe[*c*]:TSTamp:FORMat?

Parameter ***rule*** Rule for reading the timestamp data. DELTa|ABSolute (default). Parameter data type is CPD.

rule=ABS sets the returned data to the incremental value for the first timestamp data.

rule=DELT sets the returned data to the incremental value for the previous timestamp data.

Query response *rule* <newline>

rule returns the present setting of the rule, DELT or ABS. Response data type is CRD.

Example :TRAC:TST:FORM DELT

:TRAC2:TST:FORM?

TRIGger Subsystem

For the numeric suffix [*c*], see “Numeric Suffix” on page 1-8.

:ABORt<:ACQuire|:TRANsient|[:ALL]>

Aborts the specified device action for the specified channel. Trigger status is changed to idle.

Syntax

:ABORt<:ACQuire|:TRANsient|[:ALL]> [*chanlist*]

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

chanlist Channels. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Example

:ABOR:ACQ (@2)

:ARM<:ACQuire|:TRANsient|[:ALL]>[:IMMediate]

Sends an immediate arm trigger for the specified device action to the specified channel. When the status of the specified device action is initiated, the arm trigger causes a layer change from arm to trigger.

Syntax

:ARM<:ACQuire|:TRANsient|[:ALL]>[:IMMediate] [*chanlist*]

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

chanlist Channels. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Example :ARM:ACQ (@2)

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:BYPass

Enables or disables a bypass for the event detector in the arm layer.

Syntax :ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:BYPass *bypass*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:BYPass?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

bypass	Bypass setting. ONCE|OFF (default). Parameter data type is CPD.

bypass=OFF disables the bypass.

bypass=ONCE enables the bypass, but only for the first passage.

Query response *response* <newline>

response returns the present setting of the bypass, OFF or ONCE. Response data type is CRD.

Example :ARM:BYP ONCE

:ARM2:TRAN:BYP?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:COUNt

Sets the arm count for the specified device action.

Syntax :ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:COUNt *arm_count*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:COUNt? [*arm_count*]

:ARM[*c*][:ALL][:LAYer]:COUNt? *arm_count*

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

arm_count Arm count. *value* (1 to 100000 or 2147483647)|INFinity|MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf+. *value*=2147483647 indicates infinity.

Query does not support *arm_count=value* and INFinity.

Arm count × *Trigger count* must be less than 100001.

Query response

response <newline>

response returns the present setting of arm count. If a parameter is specified, *response* returns the value assigned to DEF, MIN, MAX, or INF. Response data type is NR1. If the arm count is set to infinity, *response* returns 2147483647.

Example

:ARM:COUN 10

:ARM2:TRAN:COUN?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:DELay

Sets the arm delay for the specified device action.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:DELay *delay*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:DELay? [*delay*]

:ARM[*c*][:ALL][:LAYer]:DELay? *delay*

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

delay Arm delay, in seconds. *value* (0 to 100)|MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *delay=value*.

Query response

response <newline>

response returns the present setting of arm delay. If a parameter is specified, *response* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:ARM:DEL 0.1

:ARM2:TRAN:DEL?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce:LAN

Specifies one or more LXI triggers used for the arm source for the specified device action.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce:LAN *lan_id*{,*lan_id*}

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:SOURce:LAN?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

lan_id LAN ID of the LXI trigger. LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7. All is selected as default. Parameter data type is CPD.

Query response

response <newline>

response returns the present setting, LAN0 through LAN7. Response data type is CRD. Multiple responses are separated by a comma.

Example

:ARM:SOUR:LAN LAN7

:ARM2:TRAN:SOUR:LAN?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce[:SIGNal]

Selects the arm source for the specified device action.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce[:SIGNal] *source*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:SOURce[:SIGNal]?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

source Arm source. AINT (default)|BUS|TIMer|INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12| EXT13|EXT14. Parameter data type is CPD.

source=AINT (automatic internal) automatically selects the arm source most suitable for the present operating mode by using internal algorithms.

source=BUS selects the remote interface trigger command such as the group execute trigger (GET) and the *TRG command.

source=TIMer selects a signal internally generated every interval set by the :ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TIMer command.

source=INT1 or INT2 selects a signal from the internal bus 1 or 2, respectively.

source=LAN selects the LXI trigger specified by the :ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce:LAN command.

source=EXT*n* selects a signal from the GPIO pin *n*, which is an input port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response

response <newline>

response returns the present setting of arm source, AINT, BUS, TIM, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD.

Example

:ARM:SOUR AINT

:ARM2:TRAN:SOUR?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TIMer

Sets the interval of the TIMer arm source for the specified device action.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TIMer *interval*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:TIMer? [*interval*]

:ARM[*c*][:ALL][:LAYer]:TIMer? *interval*

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

interval Interval, in seconds. *value* (1E-5 to 1E+5)|MINimum|MAXimum|DEFault (default is 1E-5). Parameter data type is NRf+. Query does not support *interval*=*value*.

Query response

response <newline>

response returns the present setting of interval. If a parameter is specified, *response* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

:ARM:TIM 2E-4

:ARM2:TRAN:TIM?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TOUTput:SIGNal

Selects the trigger output for the status change between the idle state and the arm layer. Multiple trigger output ports can be set.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TOUTput:SIGNal *output*{,*output*}

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:TOUTput:SIGNal?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

output Trigger output port. EXT1 (default)|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14|LAN|INT1|INT2. Parameter data type is CPD.

output=INT1 or INT2 selects the internal bus 1 or 2, respectively.

output=LAN selects a LAN port.

output=EXT*n* selects the GPIO pin *n*, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response

response <newline>

response returns the present setting, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD. Multiple responses are separated by a comma.

Example

:ARM:TOUT:SIGN EXT1

:ARM2:TRAN:TOUT:SIGN?

:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TOUTput[:STATe]

Enables or disables the trigger output for the status change between the idle state and the arm layer.

Syntax

:ARM[*c*]<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:TOUTput[:STATe] *mode*

:ARM[*c*]<:ACQuire|:TRANsient>[:LAYer]:TOUTput[:STATe]?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

mode Trigger output ON or OFF. 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the trigger output.

mode=0 or OFF disables the trigger output.

Query response

response <newline>

response returns 1 or 0, and indicates that the trigger output is on or off, respectively. Response data type is NR1.

Example

:ARM:TOUT 1

:ARM2:TRAN:TOUT:STAT?

:IDLE<:ACQuire|:TRANsient|[:ALL]>?

Checks the status of the specified device action for the specified channel, and waits until the status is changed to idle.

Syntax

:IDLE[*c*]<:ACQuire|:TRANsient|[:ALL]>?

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Query response

response <newline>

response returns 1 if the specified device action is in the idle state. Response data type is NR1.

Example

:IDLE2:ACQ

:INITiate[:IMMediate]<:ACQuire|:TRANsient|[:ALL]>

Initiates the specified device action for the specified channel. Trigger status is changed from idle to initiated.

Syntax

:INITiate[:IMMediate]<:ACQuire|:TRANsient|[:ALL]> [*chanlist*]

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

chanlist Channels. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Example

```
:INIT:ACQ (@2)
```

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:BYPass

Enables or disables a bypass for the event detector in the trigger layer.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:BYPass *bypass*

:TRIGger[*c*]<:ACQuire|:TRANsient>:BYPass?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

bypass Bypass setting. ONCE|OFF (default). Parameter data type is CPD.

bypass=OFF disables the bypass.

bypass=ONCE enables the bypass, but only for the first passage.

Query response

response <newline>

response returns the present setting of the bypass, OFF or ONCE. Response data type is CRD.

Example

```
:TRIG:BYP ONCE
```

:TRIG2:TRAN:BYP?

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:COUNt

Sets the trigger count for the specified device action.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:COUNt *trigger_count*

:TRIGger[*c*]<:ACQuire|:TRANsient>:COUNt? [*trigger_count*]

:TRIGger[*c*][:ALL]:COUNt? *trigger_count*

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

trigger_count Trigger count. *value* (1 to 100000)|MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf+. Query does not support *trigger_count=value*.

Arm count × *Trigger count* must be less than 100001.

Query response

response <newline>

response returns the present setting of trigger count. If a parameter is specified, *response* returns the value assigned to DEF, MIN, or MAX. Response data type is NR1.

Example

:TRIG:COUN 10

:TRIG2:TRAN:COUN?

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:DELay

Sets the trigger delay for the specified device action.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:DELay *delay*

:TRIGger[*c*]<:ACQuire|:TRANsient>:DELay? [*delay*]

:TRIGger[*c*][:ALL]:DELay? *delay*

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter ***delay*** Trigger delay, in seconds. *value* (0 to 100)|MINimum|MAXimum|DEFault (default is 0). Parameter data type is NRf+. Query does not support *delay=value*.

Query response *response* <newline>

response returns the present setting of trigger delay. If a parameter is specified, *response* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:TRIG:DEL 0.1
:TRIG2:TRAN:DEL?
```

:TRIGger<:ACQuire|:TRANsient|[:ALL]>[:IMMediate]

Sends an immediate trigger for the specified device action to the specified channel. When the status of the specified device action is initiated, the trigger causes the specified device action.

Syntax :TRIGger<:ACQuire|:TRANsient|[:ALL]>[:IMMediate] [*chanlist*]

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter ***chanlist*** Channels. Parameter data type is channel list. (@1)|(@2)|(@1,2)|(@1:2)|(@2,1)|(@2:1). See “Channel List Parameter” on page 1-8.

(@1) selects channel 1 only. (@2) selects channel 2 only. (@1,2), (@1:2), (@2,1), and (@2:1) selects both channels 1 and 2.

If this parameter is not specified, *chanlist*=(@1) is set.

Example

```
:TRIG:ACQ (@2)
```

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:SOURce:LAN

Specifies one or more LXI triggers used for the trigger source for the specified device action.

Syntax :TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:SOURce:LAN *lan_id*{,*lan_id*}

:TRIGger[*c*]<:ACQuire|:TRANsient>:SOURce:LAN?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

lan_id	LAN ID of the LXI trigger. LAN0|LAN1|LAN2|LAN3|LAN4|LAN5|LAN6|LAN7. All is selected as default. Parameter data type is CPD.

Query response

response <newline>

response returns the present setting, LAN0 through LAN7. Response data type is CRD. Multiple responses are separated by a comma.

Example

:TRIG:SOUR:LAN LAN7

:TRIG2:TRAN:SOUR:LAN?

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:SOURce[:SIGNal]

Selects the trigger source for the specified device action.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:SOURce[:SIGNal] *source*

:TRIGger[*c*]<:ACQuire|:TRANsient>:SOURce[:SIGNal]?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

source	Trigger source. AINT (default)|BUS|TIMer|INT1|INT2|LAN|EXT1|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14. Parameter data type is CPD.

source=AINT (automatic internal) automatically selects the trigger source most suitable for the present operating mode by using internal algorithms.

source=BUS selects the remote interface trigger command such as the group execute trigger (GET) and the *TRG command.

source=TIMer selects a signal internally generated every interval set by the :TRIGger<:ACQuire|:TRANsient|[:ALL]>:TIMer command.

source=INT1 or INT2 selects a signal from the internal bus 1 or 2, respectively.

source=LAN*n* selects a LXI trigger specified by the :TRIGger<:ACQuire|:TRANsient|[:ALL]>:SOURce:LAN command.

source=EXT*n* selects a signal from the GPIO pin *n*, which is an input port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response *response* <newline>

response returns the present setting of trigger source, AINT, BUS, TIM, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD.

Example

```
:TRIG:SOUR EXT1
:TRIG2:TRAN:SOUR:SIGN?
```

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:TIMer

Sets the interval of the TIMer trigger source for the specified device action.

Syntax

```
:TRIGger[c]<:ACQuire|:TRANsient|[:ALL]>:TIMer interval
:TRIGger[c]<:ACQuire|:TRANsient>:TIMer? [interval]
:TRIGger[c][:ALL]:TIMer? interval
```

For <:ACQuire|:TRANsient|[:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

interval	Interval, in seconds. *value* (1E-5 to 1E+5)|MINimum|MAXimum|DEFault (default is 1E-5). Parameter data type is NRf+. Query does not support *interval=value*.

Query response *response* <newline>

response returns the present setting of interval. If a parameter is specified, *response* returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.

Example

```
:TRIG:TIM 2E-4
:TRIG2:TRAN:TIM?
```

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:TOUTput:SIGNal

Selects the trigger output for the status change between the arm layer and the trigger layer. Multiple trigger output ports can be set.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:TOUTput:SIGNal *output*{,*output*}

:TRIGger[*c*]<:ACQuire|:TRANsient>:TOUTput:SIGNal?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

output Trigger output port. EXT1 (default)|EXT2|EXT3|EXT4|EXT5|EXT6|EXT7|EXT8|EXT9|EXT10|EXT11|EXT12|EXT13|EXT14|LAN|INT1|INT2. Parameter data type is CPD.

output=INT1 or INT2 selects the internal bus 1 or 2, respectively.

output=LAN selects a LAN port.

output=EXT*n* selects the GPIO pin *n*, which is an output port of the Digital I/O D-sub connector on the rear panel. *n*=1 to 14.

Query response

response <newline>

response returns the present setting, INT1, INT2, LAN, or EXT1 through EXT14. Response data type is CRD. Multiple responses are separated by a comma.

Example

:TRIG:TOUT:SIGN EXT3

:TRIG2:TRAN:TOUT:SIGN?

:TRIGger<:ACQuire|:TRANsient|[:ALL]>:TOUTput[:STATe]

Enables or disables the trigger output for the status change between the arm layer and the trigger layer.

Syntax

:TRIGger[*c*]<:ACQuire|:TRANsient|[:ALL]>:TOUTput[:STATe] *mode*

:TRIGger[*c*]<:ACQuire|:TRANsient>:TOUTput[:STATe]?

For <:ACQuire|:TRANsient|[:ALL]> and <:ACQuire|:TRANsient>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.

Parameter

mode Trigger output ON or OFF. 1|ON|0|OFF (default). Parameter data type is boolean.

mode=1 or ON enables the trigger output.

mode=0 or OFF disables the trigger output.

Query response

response <newline>

response returns 1 or 0, and indicates that the trigger output is on or off, respectively. Response data type is NR1.

Example

```
:TRIG:TOUT 1

:TRIG2:TRAN:TOUT:STAT?
```

5 Error Messages

This chapter shows the error code/messages returned from Agilent B2900 when any error occurred during a SCPI program is executed.

Error messages are classified by error number as listed in the following table.

Error range	Error category	Standard event status register bit
0	No error	
-100 to -199	Command error	bit5
-200 to -299	Execution error	bit4
-300 to -399	Device-dependent error	bit3
-400 to -499	Query error	bit2
1 to 32767	B2900 specific error	bit3

Negative error numbers (command error, execution error, device-dependent error, query error) are standard SCPI errors.

Positive error numbers are B2900 specific errors, not standard SCPI errors.

When B2900 is in the remote control state, the occurrence of an error (except for error number 0 or emergency error) sets the corresponding bit in the standard event status register. An emergency error sets the corresponding bit in the emergency status register.

If an error occurs, the error number and message are placed in the error queue, which can be read by the `:SYSTem:ERRor?` query command. Then the front-panel ERR indicator turns on. Errors are cleared by reading them. When all errors are read from the queue, the errors are cleared and the ERR indicator turns off. Errors are retrieved in the FIFO (first-in-first-out) order. The first error returned is the first error that was stored.

The error queue is also cleared by the common command `*CLS`, and when power is turned on. The error queue is not cleared by the `*RST` command. For these commands, see and Chapter 3.

If more errors have occurred than can fit in the buffer, the last error stored in the queue (the most recent error) is replaced with -350, “Error queue overflow”. No additional errors are stored until removing errors from the queue. If no errors have occurred when reading the error queue, the instrument responds with +0, “No error”.

No Error

This message indicates that Agilent B2900 has no errors.

Error 0 `No error`

The error queue is completely empty. Every error/event in the queue has been read or the queue was purposely cleared by power-on, `*CLS`, and so on.

Command Error

If syntax of SCPI command is *not* valid, a `-1`*XX* error occurs.

Error -100 `Command error`

Generic syntax error that cannot be determined more specifically.

Error -101 `Invalid character`

An invalid character for the type of a syntax element was received; for example, a header containing an ampersand.

Error -102 `Syntax error`

An unrecognized command or data type was received; for example, a string was received when B2900 does not accept strings.

Error -103 `Invalid separator`

An illegal character was received when a separator was expected; for example, the semicolon was omitted after a program message unit.

Error -104 `Data type error`

An improper data type was received; for example, numeric data was expected but string data was received.

Error -105 `GET not allowed`

A group execute trigger was received within a program message.

Error -108 `Parameter not allowed`

Too many parameters for the command were received.

Error -109 `Missing parameter`

Fewer parameters were received than required for the command.

Error -110 `Command header error`

An error was detected in the header. This error message is reported if B2900 cannot determine the more specific header errors -111 through -114.

Error -111 `Header separator error`

An illegal character for a header separator was received; for example, no white space between the header and parameter.

Error -112 `Program mnemonic too long`

A keyword in the command header contains more than twelve characters.

Error -113 `Undefined header`

An undefined command header was received; for example, `*XYZ`.

Error -114 `Header suffix out of range`

The value of a numeric suffix attached to a program mnemonic is out of range; for example, `:OUTP3:FILT:AUTO` specifies illegal channel number 3.

Error -120 `Numeric data error`

Numeric (including the non-decimal numeric types) data error. This error message is reported when B2900 cannot determine the more specific errors -121 through -128.

Error -121 `Invalid character in number`

An invalid character for the data type was received; for example, an alpha-character was received when the type was decimal numeric.

Error -123 `Exponent too large`

The magnitude of the exponent was larger than 32000.

Error -124 `Too many digits`

The mantissa of a decimal numeric data contained more than 255 digits excluding leading zeros.

Error -128 `Numeric data not allowed`

Numeric data is not allowed in this position for this command.

Error -130 `Suffix error`

An error was detected in the suffix. This error message is reported if B2900 cannot determine the more specific suffix errors -131 through -138.

Error -131 `Invalid suffix`

The suffix does not follow the correct syntax or the suffix is inappropriate.

Error -134 `Suffix too long`

The suffix contains more than 12 characters.

Error -138 `Suffix not allowed`

A suffix was received after a numeric parameter that does not allow suffixes.

Error -140 `Character data error`

An error was detected in a character parameter. This error message is reported if B2900 cannot determine the more specific errors -141 through -148.

Error -141 `Invalid character data`

Either the character parameter contains an invalid character or the particular element received is not valid for the command header.

Error -144 `Character data too long`

The character parameter contains more than 12 characters.

Error -148 `Character data not allowed`

A character parameter is not allowed for this position.

Error -150 `String data error`

An error was detected in a string parameter. This error is reported if B2900 cannot determine a more specific error -151 and -158.

Error -151 `Invalid string data`

An invalid string parameter data was received; for example, an END message was received before the terminal quote character.

Error -158 `String data not allowed`

A string parameter data was received but was not allowed at this point.

Error -160 `Block data error`

An error was detected in a block data. This error is reported if B2900 cannot determine more specific errors -161 and -168.

Error -161 `Invalid block data`

An invalid block data was received; for example, an END message was received before the length was satisfied.

Error -168 `Block data not allowed`

A legal block data was received but was not allowed at this point.

Error -170 `Expression error`

An error was detected in an expression. This error is reported if B2900 cannot determine more specific errors -171 and -178.

Error -171 `Invalid expression`

The expression was invalid; for example, unmatched parentheses or an illegal character.

Error -178 `Expression data not allowed`

An expression was received but was not allowed at this point.

Execution Error

Agilent B2900 reports `-2`*XX* errors when it is unable to perform a valid programming command.

Error -200 `Execution error`

Generic execution error for B2900 that cannot be determined more specifically.

Error -220 `Parameter error; Invalid channel list`

`Parameter error; Invalid group defnition`

Invalid channel list or group defnition was specified. Set appropriate value.

Error -221 `Settings conflict;` *message*`; channel` *n*

A specified parameter setting could not be executed due to the present device state. Check the settings specified by *message* and channel *n*, and set appropriate value.

Error -222 `Data out of range;` *message*`; channel` *n*

Interpreted value of the program was out of range as defined by B2900. Check the B2900 settings specified by *message* and channel *n*, and set appropriate value.

Error -223 `Too much data`

Too many parameters were sent. Reduce number of list data.

Error -224 `Illegal parameter value`

Illegal parameter value was sent. Set appropriate parameter value.

Error -230 `Data corrupt or stale`

Possibly invalid data; new reading started but not completed since last access.

Error -231 `Data questionable`

Measurement accuracy is suspect.

Error -232 `Invalid format`

The data format or structure is inappropriate.

Error -233 `Invalid version`

The version of the data format is incorrect to the instrument.

Error -240 `Hardware error`

A hardware problem in B2900. This error message is reported if B2900 cannot detect the more specific error -241.

Error -241 `Hardware missing; To recover channel, execute *TST?`

A program command or query could not be executed because of missing hardware; for example, an option was not installed. Execute the `*TST?` command to recover or unlock channel.

Device-Dependent Error

-3*XX* errors indicate that Agilent B2900 has detected an error that is not a command error, a query error, or an execution error; some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors.

Error -300 `Device-specific error`

Generic device-dependent error for B2900 that cannot be determined more specifically.

Error -310 `System error`

Some error, termed “system error” by B2900, has occurred.

Error -311 `Memory error`

An error was detected in B2900’s memory.

Error -313 `Calibration memory lost; Calibration data has been lost, Calibration data is initialized; Channel`*n*

`Calibration memory lost; Nonvolatile data saved by the *CAL? command has been lost; Channel`*n*

Non-volatile data related to the `*CAL?` command has been lost.

Error -315 `Configuration memory lost`

Non-volatile configuration data saved by B2900 has been lost.

Error -321 `Out of memory`

Too many data was sent at a time.

Error -350 `Queue overflow`

This code is entered into the queue instead of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.

Query Error

If the output queue control of Agilent B2900 detects one of following problems, a *-4XX* error occurs.

- An attempt was made to read data from the output queue when no output data is present or pending.
- Data in the output queue has been lost.

Error -400 `Query error`

Generic query error for B2900 that cannot be determined more specifically.

Error -410 `Query INTERRUPTED`

A condition causing an INTERRUPTED query error occurred; for example, a query followed by DAB or GET before a response was completely sent.

Error -420 `Query UNTERMINATED`

A condition causing an UNTERMINATED query error occurred; for example, B2900 was addressed to talk and an incomplete program message was received.

Error -430 `Query DEADLOCKED`

A condition causing a DEADLOCKED query error occurred; for example, both input buffer and output buffer are full and B2900 cannot continue.

Error -440 `Query UNTERMINATED after indefinite response`

A query was received in the same program message after a query requesting an indefinite length response was executed.

B2900 Specific Error

Positive error numbers are Agilent B2900 specific errors, not standard SCPI errors. Consult service for errors 111 to 140.

Error 111 `Self-calibration failed; Voltage offset,` *item*`; channel` *n*

Failed the voltage offset self-calibration specified by *item* and channel *n*.

Error 112 `Self-calibration failed; Current offset,` *item*`; channel` *n*

Failed the current offset self-calibration specified by *item* and channel *n*.

Error 113 `Self-calibration failed; Voltage gain,` *item*`; channel` *n*

Failed the voltage gain self-calibration specified by *item* and channel *n*.

Error 114 `Self-calibration failed; Current gain,` *item*`; channel` *n*

Failed the current gain self-calibration specified by *item* and channel *n*.

Error 115 `Self-calibration failed; CMR DAC,` *item*`; channel` *n*

Failed the CMR DAC self-calibration specified by *item* and channel *n*.

Error 121 `Self-test failed; CPU communication,` *item*`; channel` *n*

Failed the CPU communication test specified by *item* and channel *n*.

Error 122 `Self-test failed; Fan status,` *item*`; channel` *n*

Failed the fan status test specified by *item* and channel *n*.

Error 131 `Self-test failed; SMU communication,` *item*`; channel` *n*

Failed the SMU communication test specified by *item* and channel *n*.

Error 132 `Self-test failed; CPLD access,` *item*`; channel` *n*

Failed the CPLD access test specified by *item* and channel *n*.

Error 133 `Self-test failed; Trigger count,` *item*`; channel` *n*

Failed the trigger count test specified by *item* and channel *n*.

Error 134 `Self-test failed; DAC/ADC,` *item*`; channel` *n*

Failed the DAC/ADC test specified by *item* and channel *n*.

Error 135 `Self-test failed; Loop control,` *item*`; channel` *n*

Failed the loop control test specified by *item* and channel *n*.

Error 136 `Self-test failed; I sense,` *item*`; channel` *n*

Failed the current sense test specified by *item* and channel *n*.

Error 137 `Self-test failed; V sense,` *item*`; channel` *n*

Failed the voltage sense test specified by *item* and channel *n*.

Error 138 `Self-test failed; F-COM comparison,` *item*`; channel` *n*

Failed the F-COM comparison test specified by *item* and channel *n*.

Error 139 `Self-test failed; V switch,` *item*`; channel` *n*

Failed the voltage switch test specified by *item* and channel *n*.

Error 140 `Self-test failed; Temperature sensor,` *item*`; channel` *n*

Failed the temperature sensor test specified by *item* and channel *n*.

Error 202 `Instrument locked by another I/O session`

The requested operation is not allowed because the instrument is locked by another I/O session. The instrument must be unlocked.

Error 210 `Operation is not completed`

Operation is still in progress. Wait for operation complete.

Error 211 `Cannot switch low sense terminal with output on`

Output relay must be off to switch low sense terminal.

Error 212 `Output relay must be on`

Output relay must be on to perform the requested operation.

Error 213 `Output relay must be off`

Output relay must be off to perform the requested operation.

Error 214 `Display must be enabled`

Display is currently disabled. Set remote display on.

Error 215 `Remote sensing must be on`

Remote sensing must be on to perform the requested operation.

Error 216 `Auto resistance measurement must be off`

Automatic resistance measurement must be off to perform the requested operation.

Error 301 `Emergency; Overvoltage status detected; Channel`*n*

Overvoltage status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 302 `Emergency; Overcurrent status(245 V) detected; Channel`*n*

overcurrent status (245 V) was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 303 `Emergency; Overcurrent status(35 V) detected; Channel`*n*

overcurrent status (35 V) was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 304 `Emergency; Over range current status detected; Channel`*n*

Over range current status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 305 `Emergency; High temperature1 status detected; Channel`*n*

High temperature 1 status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 306 `Emergency; High temperature2 status detected; Channel`*n*

High temperature 2 status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 307 `Emergency; High temperature3 status detected; Channel`*n*

High temperature 3 status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 308 `Emergency; High temperature4 status detected; Channel` *n*

High temperature 4 status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 311 `Emergency; Abuse detected; Channel` *n*

Abuse status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 312 `Emergency; F-COM(minus) abuse detected; Channel` *n*

F-COM (minus) status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 313 `Emergency; F-COM(plus) abuse detected; Channel` *n*

F-COM (plus) abuse status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 314 `Emergency; Low sense(minus) abuse detected; Channel` *n*

Low sense (minus) abuse status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 315 `Emergency; Low sense(plus) abuse detected; Channel` *n*

Low sense (plus) abuse status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 321 `Emergency; SMU main power supply failure detected; Channel` *n*

SMU main power supply failure was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 322 `Emergency; SMU positive power supply failure detected; Channel` *n*

SMU positive power supply failure was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 323 `Emergency; SMU negative power supply failure detected; Channel` *n*

SMU negative power supply failure was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 324 `Emergency; SMU power supply was turned off; Channel`*n*

SMU power supply was turned off because emergency status was detected in channel *n*. All channels were disabled. Execute the *TST? command.

Error 331 `Emergency; Interlock open detected`

Interlock open was detected. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command. Do not open interlock circuit while SMU is in high voltage state.

Error 341 `Emergency; Fan speed is too slow`

Too slow fan speed status was detected. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 342 `Emergency; Fan speed is too fast`

Too fast fan speed status was detected. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 351 `Emergency; Internal communication failure detected by SMU; Channel`*n*

Internal communication failure was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 352 `Emergency; Watchdog timer expired; Channel`*n*

Watchdog timer expired status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 353 `Emergency; F-COM CPLD reset detected; Channel`*n*

F-COM CPLD reset status was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 354 `Emergency; VADC data was lost; Channel`*n*

Channel *n* voltage ADC data was lost. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 355 `Emergency; IADC data was lost; Channel`*n*

Channel *n* current ADC data was lost. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 356 `Emergency; Sense data FIFO overflow detected; Channel` *n*

Sense data FIFO overflow was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 361 `Emergency; Internal communication failure detected by CPU; Channel` *n*

Channel *n* internal communication failure was detected by CPU. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 362 `Emergency; Internal command queue overflow detected; Channel` *n*

Internal command queue overflow was detected in channel *n*. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 363 `Emergency; Sense data was not received for acquire trigger; Channel` *n*

Channel *n* sense data was not received for acquire trigger. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 364 `Emergency; Unexpected sense data was received; Channel` *n*

Unexpected sense data was received. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 365 `Emergency; Sense data was not received in Timer period; Channel` *n*

Data communication failure. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 366 `Emergency; Timestamp FIFO overflow detected; Channel` *n*

Data communication failure. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.

Error 700 `ProgramMemory; Program size overflow`

Program memory cannot save the program. Reduce program size.

Error 701 `ProgramMemory; Invalid variable`

Appropriate variable name must be specified.

Error 702 `ProgramMemory; Invalid variable number`

Appropriate variable name must be specified.

Error 703 `ProgramMemory; Query command is not supported`

Memory program cannot contain query command.

Error 704 `ProgramMemory; Program is not selected`

Appropriate program name must be specified.

Error 705 `ProgramMemory; Cannot execute program while another program is running`

Another program is running. Execute the program after it is stopped.

Error 706 `ProgramMemory; Cannot execute program while this program is running`

This program is running. Execute the program after it is stopped.

Error 707 `ProgramMemory; Cannot step program while program is running`

Program is running. Step execution is effective when program is paused or stopped.

Error 708 `ProgramMemory; Cannot continue program while program is running`

Program is running. Program continue is effective when program is paused.

Error 709 `ProgramMemory; Cannot continue program while program is stopped`

Program is stopped. Program continue is effective when program is paused.

Error 710 `ProgramMemory; Program line is too long`

Program memory cannot save the program. Reduce program line.

Error 711 `ProgramMemory; Variable length is too long`

Variable contains too many data. Reduce variable length.

Error 712 `ProgramMemory; Unsupported command is used in program`

Memory program cannot contain the specified command.

Error 713 `ProgramMemory; Cannot set multiple INIT commands in program line`

A program line cannot contain multiple INIT commands.

Error 714 `ProgramMemory; Invalid character in program line`

Program line contains invalid character. Use appropriate characters.

Error 715 `ProgramMemory; Invalid character in program name`

Appropriate program name must be specified.

Error 716 `ProgramMemory; Program count overflow`

Program memory cannot save the program. Delete dispensable program.

Error 801 `Calculate; Expression list full`

Cannot save the expression. Delete dispensable expression.

Error 802 `Calculate; Expression cannot be deleted`

Cannot delete the specified expression. Specify erasable expression.

Error 803 `Calculate; Missmatched parenthesis`

Number of open and close parentheses must be the same.

Error 804 `Calculate; Not a number of data handle`

Expression contains invalid floating point number or symbol. Enter appropriate expression. Available symbols are VOLT, CURR, RES, TIME, and SOUR.

Error 805 `Calculate; Mismatched brackets`

Number of open and close brackets must be the same.

Error 806 `Calculate; Entire expression not parsed`

Expression is not correct. Enter appropriate expression.

Error 807 `Calculate; Not an operator or number`

Expression contains not an operator or not a number. Enter appropriate expression.

Error 811 `Calculate; Error parsing value`

Expression contains invalid floating point number. Enter appropriate expression.

Error 812 `Calculate; Invalid data handle index`

Vector expression contains invalid index value of an array. Enter appropriate expression.

Error 813 `Calculate; Divided by zero`

Denominator must not be zero. Enter appropriate expression.

Error 814 `Calculate; Log of zero`

Expression cannot contain log 0. Enter appropriate expression.

Error 815 `Calculate; Invalid binary format string is used`

Data contains invalid binary format string. Enter appropriate expression.

Error 816 `Calculate; Invalid hex format string is used`

Data contains invalid hex format string. Enter appropriate expression.

Error 817 `Calculate; Invalid channel number is used`

Expression contains invalid channel number. Enter appropriate expression.

Error 818 `Calculate; Null expression`

Expression is not defined. Enter appropriate expression.

Error 819 `Calculate; Null expression in parentheses`

Expression contains empty parentheses. Enter appropriate expression.

Error 820 `Calculate; Null expression in brackets`

Expression contains empty brackets. Enter appropriate expression.

Error 821 `Calculate; Fed disabled MATH for limit test`

Limit test tried to feed the math result currently disabled. Enable the math expression.

Error 822 `Calculate; Missmatched trigger counts`

Trigger count of grouped channels must be the same.

Error 823 `Calculate; Missmatched vector lengths`

Vector length of grouped channels must be the same.

Error 824 `Calculate; Invalid character in math name`

Appropriate math expression name must be specified.

Error 861 `Trace; Illegal with storage active`

Storage device must be idle to perform the requested operation.

Error 862 `Trace; No trace data`

Trace buffer must contain data to perform the requested operation.

Error 870 `Macro file size error`

Macro file size error. Reduce file size.

Error 871 `Cannot create data on non-volatile memory`

Cannot save data to non-volatile memory.

Error 900 `Internal system error`

Internal system error is detected.

Error 950 `Unsupported parameter`

Conventional command set error. Specified parameter is not supported by B2900.

Error 951 `Unsupported command`

Conventional command set error. Specified command is not supported by B2900.

Error 952 `Cannot disable the state`

Conventional command set error. B2900 cannot disable the state at this time.

Error 953 `Cannot enable the state`

Conventional command set error. B2900 cannot enable the state at this time.

Error 954 `Cannot send binary data via SOCKET interface`

Conventional command set error. B2900 cannot send binary data via SOCKET interface at this time.

6 Using Your Existing Programs

Agilent B2900 supports two command sets, default and conventional. The default command set is designed to support all B2900 functions. The conventional command set is designed for using existing programs for controlling existing instruments, such as Series 2400 from Keithley Instruments, Inc.

To switch the command set

To switch to the conventional command set, enter the :SYSTem:LANGuage “2400” command or use the front panel keys System>SCPI>2400.

To return to the default command set, enter the :SYSTem:LANGuage “DEF” command or use the front panel keys System>SCPI>Default.

B2900 will restart automatically, and the command set will be switched after the boot.

To use default command set

The default command set must be used for normal programming which uses all B2900 functions. See the previous chapters which describe B2900’s default SCPI commands. You do not need to read this chapter.

To use conventional command set

The conventional command set can be used for using programs created for existing instruments. Refer to the following procedure and edit or modify your program as needed. Also see the existing instrument’s manuals for its SCPI commands.

1. Open your program.
2. Check the SCPI commands used in the program. Confirm that the commands are supported by B2900. See the following sections.
   - “Conventional commands supported by B2900”
   - “Conventional commands partially supported by B2900”
   - “Conventional commands not supported by B2900”
3. Do not change the command if it is supported by B2900.
4. Delete or comment-out the command if it is not supported by B2900. B2900 will generate an error if an unsupported command is still effective.
5. Modify the command if it is partially supported by B2900. B2900 will support a part of some functions. Or, delete or comment-out the command if it is unnecessary.
6. Save as a new program.

Conventional commands supported by B2900

:ABORt

:ARM[:SEQuence[1]][:LAYer[1]]:COUNt <NRf|DEFault|MINimum|MAXimum>

:ARM[:SEQuence[1]][:LAYer[1]]:COUNt?

:ARM[:SEQuence[1]][:LAYer[1]]:SOURce?

:ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure]:DIRection SOURce|ACCeptor

:ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure]:DIRection?

:ARM[:SEQuence[1]][:LAYer[1]]:TIMer <NRf>

:ARM[:SEQuence[1]][:LAYer[1]]:TIMer?

:CALCulate[1]:DATA:LATest?

:CALCulate[1]:DATA?

:CALCulate[1]:MATH[:EXPRession]:CATalog?

:CALCulate[1]:MATH[:EXPRession][:DEFine] <form>

:CALCulate[1]:MATH[:EXPRession][:DEFine]?

:CALCulate[1]:MATH[:EXPRession]:DELete:ALL

:CALCulate[1]:MATH[:EXPRession]:DELete[:SELected] <SPD>

:CALCulate[1]:MATH[:EXPRession]:NAME <SPD>

:CALCulate[1]:MATH[:EXPRession]:NAME?

:CALCulate[1]:MATH:UNITs <name>

:CALCulate[1]:MATH:UNITs?

:CALCulate[1]:STATe <Bool>

:CALCulate[1]:STATe?

:CALCulate2:CLIMits:BCONtrol IMMediate|END

:CALCulate2:CLIMits:BCONtrol?

:CALCulate2:CLIMits:CLEar:AUTO <Bool>

:CALCulate2:CLIMits:CLEar:AUTO?

:CALCulate2:CLIMits:CLEar[:IMMediate]

:CALCulate2:CLIMits:FAIL:SOURce2 <NRf|NDN>

:CALCulate2:CLIMits:FAIL:SOURce2?

:CALCulate2:CLIMits:MODE GRADing|SORTing

:CALCulate2:CLIMits:MODE?

:CALCulate2:CLIMits:PASS:SOURce2 <NRf|NDN>

:CALCulate2:CLIMits:PASS:SOURce2?

:CALCulate2:DATA:LATest?

:CALCulate2:DATA?

:CALCulate2:FEED CALCulate[1]|VOLTage|CURRent|RESistance

:CALCulate2:FEED?

:CALCulate2:LIMit[1]:COMPliance:FAIL IN|OUT

:CALCulate2:LIMit[1]:COMPliance:FAIL?

:CALCulate2:LIMit[1]:COMPliance:SOURce2 <NRf|NDN>

:CALCulate2:LIMit[1]:COMPliance:SOURce2?

:CALCulate2:LIMit[1]:FAIL?

:CALCulate2:LIMit[1]:STATe <Bool>

:CALCulate2:LIMit[1]:STATe?

:CALCulate2:LIMit[2|3|5-12]:FAIL?

:CALCulate2:LIMit[2|3|5-12]:LOWer[:DATA] <NRf>

:CALCulate2:LIMit[2|3|5-12]:LOWer[:DATA]? [DEFault|MINimum|MAXimum]

:CALCulate2:LIMit[2|3|5-12]:LOWer:SOURce2 <NRf|NDN>

:CALCulate2:LIMit[2|3|5-12]:LOWer:SOURce2?

:CALCulate2:LIMit[2|3|5-12]:PASS:SOURce2 <NRf|NDN>

:CALCulate2:LIMit[2|3|5-12]:PASS:SOURce2?

:CALCulate2:LIMit[2|3|5-12]:STATe <Bool>

:CALCulate2:LIMit[2|3|5-12]:STATe?

:CALCulate2:LIMit[2|3|5-12]:UPPer[:DATA]
<NRf|DEFault|MINimum|MAXimum>

:CALCulate2:LIMit[2|3|5-12]:UPPer[:DATA]? [DEFault|MINimum|MAXimum]

:CALCulate2:LIMit[2|3|5-12]:UPPer:SOURce2 <NRf|NDN>

:CALCulate2:LIMit[2|3|5-12]:UPPer:SOURce2?

:CALCulate2:LIMit4:SOURce2 <NRf>

:CALCulate2:LIMit4:SOURce2?

:CALCulate2:NULL:ACQuire

:CALCulate2:NULL:OFFSet <NRf|DEFault|MINimum|MAXimum>

:CALCulate2:NULL:OFFSet? [DEFault|MINimum|MAXimum]

:CALCulate2:NULL:STATe <Bool>

:CALCulate2:NULL:STATe?

:CALCulate3:DATA?

:CALCulate3:FORMat MEAN|SDEViation|MAXimum|MINimum|PKPK

:CALCulate3:FORMat?

:CONFigure:CURRent[:DC]

:CONFigure:RESistance

:CONFigure:VOLTage[:DC]

:CONFigure?

:DISPlay:CNDisplay

:DISPlay:DIGits <4|5|6|7|DEFault|MINimum|MAXimum>

:DISPlay:DIGits? [DEFault|MINimum|MAXimum]

:DISPlay:ENABle <Bool>

:DISPlay:ENABle?

:DISPlay[:WINDow[1]]:TEXT:DATA <SPD>

:DISPlay[:WINDow[1]]:TEXT:DATA?

:DISPlay[:WINDow[1]]:TEXT:STATe <Bool>

:DISPlay[:WINDow[1]]:TEXT:STATe?

:DISPlay:WINDow2:TEXT:DATA <SPD>

:DISPlay:WINDow2:TEXT:DATA?

:DISPlay:WINDow2:TEXT:STATe <Bool>

:DISPlay:WINDow2:TEXT:STATe?

:FETCh?

:FORMat:BORDer <NORMal|SWAPped>

:FORMat:BORDer?

:FORMat:ELEMents:CALCulate <CALC|TIME|STATus>

:FORMat:ELEMents:CALCulate?

:FORMat:ELEMents[:SENSe[1]]
<VOLTage|CURRent|RESistance|TIME|STATus>

:FORMat:ELEMents[:SENSe[1]]?

:FORMat:SOURce2 <ASCii|HEXadecimal|OCTal|BINary>

:FORMat:SOURce2?

:FORMat:SREGister <ASCii|HEXadecimal|OCTal|BINary>

:FORMat:SREGister?

:FORMat[:DATA] <ASCii|REAL|SREal>[,NRf]

:FORMat[:DATA]?

:INITiate[:IMMediate]

:MEASure:CURRent[:DC]?

:MEASure:RESistance?

:MEASure:VOLTage[:DC]?

:MEASure?

:OUTPut[1]:INTerlock:TRIPped?

:OUTPut[1]:SMODe <HIMPedance|NORMal|ZERO|GURAd>

:OUTPut[1]:SMODe?

:OUTPut[1][:STATe] <Bool>

:OUTPut[1][:STATe]?

:READ?

[:SENSe[1]]:CURRent[:DC]:NPLCycles <NRf|DEFault|MINimum|MAXimum>

[:SENSe[1]]:CURRent[:DC]:NPLCycles? [DEFault|MINimum|MAXimum]

[:SENSe[1]]:CURRent[:DC]:PROTection[:LEVel] <NRf|DEFault|MINimum|MAXimum>

[:SENSe[1]]:CURRent[:DC]:PROTection[:LEVel]? DEFault|MINimum|MAXimum

[:SENSe[1]]:CURRent[:DC]:PROTection:TRIPped?

[:SENSe[1]]:CURRent[:DC]:RANGe:AUTO <Bool>

[:SENSe[1]]:CURRent[:DC]:RANGe:AUTO?

[:SENSe[1]]:CURRent[:DC]:RANGe:AUTO:LLIMit <NRf>

[:SENSe[1]]:CURRent[:DC]:RANGe:AUTO:LLIMit?

[:SENSe[1]]:CURRent[:DC]:RANGe:AUTO:ULIMit?

[:SENSe[1]]:CURRent[:DC]:RANGe[:UPPer] <NRf|DEFault|MINimum|MAXimum|UP|DOWN>

[:SENSe[1]]:CURRent[:DC]:RANGe[:UPPer]? [DEFault|MINimum|MAXimum]

[:SENSe[1]]:DATA[:LATest]?

[:SENSe[1]]:FUNCtion:CONCurrent <Bool>

[:SENSe[1]]:FUNCtion:CONCurrent?

[:SENSe[1]]:FUNCtion:OFF <CURRent[:DC]|VOLTage[:DC]|RESistance>,..

[:SENSe[1]]:FUNCtion:OFF:ALL

[:SENSe[1]]:FUNCtion:OFF:COUNt?

[:SENSe[1]]:FUNCtion:OFF?

[:SENSe[1]]:FUNCtion[:ON] <CURRent[:DC]|VOLTage[:DC]|RESistance>,..

[:SENSe[1]]:FUNCtion[:ON]:ALL

[:SENSe[1]]:FUNCtion[:ON]:COUNt?

[:SENSe[1]]:FUNCtion[:ON]?

[:SENSe[1]]:FUNCtion:STATe? <“CURRent[:DC]”|“VOLTage[:DC]”|“RESistance”>

[:SENSe[1]]:RESistance:MODE <MANual|AUTO>

[:SENSe[1]]:RESistance:MODE?

[:SENSe[1]]:RESistance:NPLCycles <NRf|DEFault|MINimum|MAXimum>

[:SENSe[1]]:RESistance:NPLCycles? [DEFault|MINimum|MAXimum]

[:SENSe[1]]:RESistance:OCOMpensated <Bool>

[:SENSe[1]]:RESistance:OCOMpensated?

[:SENSe[1]]:RESistance:RANGe:AUTO <Bool>

[:SENSe[1]]:RESistance:RANGe:AUTO?

[:SENSe[1]]:RESistance:RANGe:AUTO:LLIMit <NRf>

[:SENSe[1]]:RESistance:RANGe:AUTO:LLIMit?

[:SENSe[1]]:RESistance:RANGe:AUTO:ULIMit <NRf>

[:SENSe[1]]:RESistance:RANGe:AUTO:ULIMit?

[:SENSe[1]]:RESistance:RANGe[:UPPer] <NRf|DEFault|MINimum|MAXimum|UP|DOWN>

[:SENSe[1]]:RESistance:RANGe[:UPPer]? [DEFault|MINimum|MAXimum]

[:SENSe[1]]:VOLTage[:DC]:NPLCycles <NRf|DEFault|MINimum|MAXimum>

[:SENSe[1]]:VOLTage[:DC]:NPLCycles? [DEFault|MINimum|MAXimum]

[:SENSe[1]]:VOLTage[:DC]:PROTection[:LEVel] <NRf|DEFault|MINimum|MAXimum>

[:SENSe[1]]:VOLTage[:DC]:PROTection[:LEVel]? <DEFault|MINimum|MAXimum>

[:SENSe[1]]:VOLTage[:DC]:PROTection:TRIPped?

[:SENSe[1]]:VOLTage[:DC]:RANGe:AUTO <Bool>

[:SENSe[1]]:VOLTage[:DC]:RANGe:AUTO?

[:SENSe[1]]:VOLTage[:DC]:RANGe:AUTO:LLIMit <NRf>

[:SENSe[1]]:VOLTage[:DC]:RANGe:AUTO:LLIMit?

[:SENSe[1]]:VOLTage[:DC]:RANGe:AUTO:ULIMit?

[:SENSe[1]]:VOLTage[:DC]:RANGe[:UPPer] <NRf|DEFault|MINimum|MAXimum|UP|DOWN>

[:SENSe[1]]:VOLTage[:DC]:RANGe[:UPPer]? [DEFault|MINimum|MAXimum]

:SOURce[1]:CLEar:AUTO <Bool>

:SOURce[1]:CLEar:AUTO?

:SOURce[1]:CLEar[:IMMediate]

:SOURce[1]:CURRent:CENTer <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent:CENTer? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent[:LEVel][:IMMediate][:AMPLitude] <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent[:LEVel][:IMMediate][:AMPLitude]? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent[:LEVel]:TRIGgered[:AMPLitude] <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent[:LEVel]:TRIGgered[:AMPLitude]? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent:MODE <FIXed|LIST|SWEep>

:SOURce[1]:CURRent:MODE?

:SOURce[1]:CURRent:RANGe <NRf|DEFault|MINimum|MAXimum|UP|DOWN>

:SOURce[1]:CURRent:RANGe? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent:RANGe:AUTO <Bool>

:SOURce[1]:CURRent:RANGe:AUTO?

:SOURce[1]:CURRent:SPAN <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent:SPAN? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent:STARt <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent:STARt? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent:STEP <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent:STEP? [DEFault|MINimum|MAXimum]

:SOURce[1]:CURRent:STOP <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:CURRent:STOP? [DEFault|MINimum|MAXimum]

:SOURce[1]:DELay <NRf|MINimum|MAXimum|DEFault>

:SOURce[1]:DELay? [MINimum|MAXimum|DEFault]

:SOURce[1]:DELay:AUTO <Bool>

:SOURce[1]:DELay:AUTO?

:SOURce[1]:FUNCtion[:MODE]?

:SOURce[1]:LIST:CURRent <NRf>{,<NRf>}..

:SOURce[1]:LIST:CURRent?

:SOURce[1]:LIST:CURRent:APPend <NRf>{,<NRf>}..

:SOURce[1]:LIST:CURRent:POINts?

:SOURce[1]:LIST:CURRent:STARt <NRf>

:SOURce[1]:LIST:CURRent:STARt?

:SOURce[1]:LIST:VOLTage <NRf>{,<NRf>}..

:SOURce[1]:LIST:VOLTage?

:SOURce[1]:LIST:VOLTage:APPend <NRf>{,<NRf>}..

:SOURce[1]:LIST:VOLTage:POINts?

:SOURce[1]:LIST:VOLTage:STARt <NRf>

:SOURce[1]:LIST:VOLTage:STARt?

:SOURce[1]:SWEep:DIRection <UP|DOWN>

:SOURce[1]:SWEep:DIRection?

:SOURce[1]:SWEep:POINts <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:SWEep:POINts? [DEFault|MINimum|MAXimum]

:SOURce[1]:SWEep:RANGing <BEST|AUTO|FIXed>

:SOURce[1]:SWEep:RANGing?

:SOURce[1]:SWEep:SPACing <LINear|LOGarithmic>

:SOURce[1]:SWEep:SPACing?

:SOURce[1]:VOLTage:CENTer <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:CENTer? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage[:LEVel][:IMMediate][:AMPLitude] <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage[:LEVel][:IMMediate][:AMPLitude]? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage[:LEVel]:TRIGgered[:AMPLitude] <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage[:LEVel]:TRIGgered[:AMPLitude]? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:MODE <FIXed|LIST|SWEep>

:SOURce[1]:VOLTage:MODE?

:SOURce[1]:VOLTage:PROTection[:LEVel] <NRf|NONE|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:PROTection[:LEVel]? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:PROTection:TRIPped?

:SOURce[1]:VOLTage:RANGe <NRf|DEFault|MINimum|MAXimum|UP|DOWN>

:SOURce[1]:VOLTage:RANGe? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:RANGe:AUTO <Bool>

:SOURce[1]:VOLTage:RANGe:AUTO?

:SOURce[1]:VOLTage:SPAN <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:SPAN? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:STARt <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:STARt? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:STEP <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:STEP? [DEFault|MINimum|MAXimum]

:SOURce[1]:VOLTage:STOP <NRf|DEFault|MINimum|MAXimum>

:SOURce[1]:VOLTage:STOP? [DEFault|MINimum|MAXimum]

:SOURce2:BSIZe <3|4>

:SOURce2:BSIZe?

:SOURce2:CLEar:AUTO <Bool>

:SOURce2:CLEar:AUTO?

:SOURce2:CLEar:AUTO:DELay <NRf|DEFault|MINimum|MAXimum>

:SOURce2:CLEar:AUTO:DELay? [DEFault|MINimum|MAXimum]

:SOURce2:CLEar[:IMMEdiate]

:SOURce2:TTL[:LEVel]:ACTual?

:SOURce2:TTL[:LEVel][:DEFault] <NRf|NDN>

:SOURce2:TTL[:LEVel][:DEFault]?

:SOURce2:TTL4:BSTate <Bool>

:SOURce2:TTL4:BSTate?

:SOURce2:TTL4:MODE <EOTest|BUSY>

:SOURce2:TTL4:MODE?

:STATus:MEASurement:CONDition?

:STATus:MEASurement:ENABle <NDN|NRf>

:STATus:MEASurement:ENABle?

:STATus:MEASurement[:EVENt]?

:STATus:OPERation:CONDition?

:STATus:OPERation:ENABle <NDN|NRf>

:STATus:OPERation:ENABle?

:STATus:OPERation[:EVENt]?

:STATus:PRESet

:STATus:QUEStionable:CONDition?

:STATus:QUEStionable:ENABle <NDN|NRf>

:STATus:QUEStionable:ENABle?

:STATus:QUEStionable[:EVENt]?

:STATus:QUEue:CLEar

:STATus:QUEue[:NEXT]?

:SYSTem:BEEPer:STATe <Bool>

:SYSTem:BEEPer:STATe?

:SYSTem:CLEar

:SYSTem:ERRor:ALL?

:SYSTem:ERRor:CODE:ALL?

:SYSTem:ERRor:CODE[:NEXT]?

:SYSTem:ERRor:COUNt?

:SYSTem:ERRor[:NEXT]?

:SYSTem:LFRequency <50|60>

:SYSTem:LFRequency?

:SYSTem:POSetup <RST|PRESet|SAV0|SAV1|SAV2|SAV3|SAV4>

:SYSTem:POSetup?

:SYSTem:PRESet

:SYSTem:RSENse <Bool>

:SYSTem:RSENse?

:SYSTem:TIME?

:SYSTem:TIME:RESet

:SYSTem:TIME:RESet:AUTO <Bool>

:SYSTem:TIME:RESet:AUTO?

:SYSTem:VERSion?

:TRACe:CLEar

:TRACe:DATA?

:TRACe:FEED <SENSe[1]|CALCulate[1]|CALCulate2>

:TRACe:FEED?

:TRACe:FEED:CONTrol <NEXT|NEVer>

:TRACe:FEED:CONTrol?

:TRACe:FREE?

:TRACe:POINts <NR1|MINimum|MAXimum|DEFault>

:TRACe:POINts? [MINimum|MAXimum|DEFault]

:TRACe:POINts:ACTual?

:TRACe:TSTamp:FORMat <ABSolute|DELTa>

:TRACe:TSTamp:FORMat?

:TRIGger[:SEQuence[1]]:COUNt <NRf|DEFault|MINimum|MAXimum>

:TRIGger[:SEQuence[1]]:COUNt? [DEFault|MINimum|MAXimum]

:TRIGger[:SEQuence[1]]:DELay <NRf|DEFault|MINimum|MAXimum>

:TRIGger[:SEQuence[1]]:DELay? [DEFault|MINimum|MAXimum]

:TRIGger[:SEQuence[1]]:SOURce?

:TRIGger[:SEQuence[1]][:TCONfigure]:DIRection <SOURce|ACCeptor >

:TRIGger[:SEQuence[1]][:TCONfigure]:DIRection?

Conventional commands partially supported by B2900

- May need to change the parameter values.

  :SYSTem:BEEPer[:IMMediate] <NRf>,<NRf>

- Parameters TLINk and BSTest do not work. No error.

  :ARM[:SEQuence[1]][:LAYer[1]]:SOURce
  <IMMediate|TLINk|TIMer|MANual|BUS|NSTest|PSTest|BSTest>

  :TRIGger[:SEQuence[1]]:SOURce
  <IMMediate|TLINk|TIMer|MANual|BUS|NSTest|PSTest|BSTest>

- Ignored. Causes no action or response.

  :ROUTe:TERMinals <FRONt|REAR>

  [:SENSe[1]]:AVERage:COUNt <NRf|DEFault|MINimum|MAXimum>

  [:SENSe[1]]:AVERage:TCONtrol <REPeat|MOVing>

  :SOURce[1]:PULSe:DELay <NRf>

  :SOURce[1]:PULSe:DELay?

  :SOURce[1]:PULSe:WIDTh <NRf>

  :SOURce[1]:PULSe:WIDTh?

  :SYSTem:AZERo:CACHing:REFResh

  :SYSTem:AZERo:CACHing:RESet

  :SYSTem:AZERo:CACHing[:STATe] <Bool>

  :SYSTem:AZERo:STATe <Bool>

  :SYSTem:CCHeck:RESistance <NRf>

  :SYSTem:GUARd <OHMS|CABLe>

  :SYSTem:KEY <NR1>

  :SYSTem:LOCal

  :SYSTem:MEMOry:INITialize

  :SYSTem:MEP:HOLDoff

:SYSTem:RWLock <Bool>

:TRIGger:CLEAr

:TRIGger:SEQuence2:SOURce <name>

:TRIGger:SEQuence2:TOUT <NRf|DEFault|MINimum|MAXimum>

- Always returns the specific value.

  :ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:ILINe? (returns 1)

  :ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:OLINe? (returns 2)

  :ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:OUTPut? (returns NONE)

  :CALCulate2:CLIMits:FAIL:SMLocation? (returns NEXT)

  :CALCulate2:CLIMits:PASS:SMLocation? (returns NEXT)

  :DISPlay[:WINDow[1]]:ATTRibutes? (returns 20 zeros)

  :DISPlay:WINDow2:ATTRibutes? (returns 32 zeros)

  :OUTPut[1]:INTerlock:STATe? (returns 1)

  :ROUTe:TERMinals? (returns FRON)

  [:SENSe[1]]:AVERage:COUNt? [DEFault|MINimum|MAXimum] (returns 10)

  [:SENSe[1]]:AVERage:TCONtrol? (retruns REP)

  [:SENSe[1]]:AVERage:STATe? (returns 1)

  [:SENSe[1]]:CURRent[:DC]:PROTection:RSYNchronize? (returns 1)

  [:SENSe[1]]:VOLTage[:DC]:PROTection:RSYNchronize? (returns 1)

  :SOURce[1]:CLEar:AUTO:MODE? (returns TCO)

  :SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACtor? (returns 1.0)

  :SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACtor:STATe? (returns 0)

  :SOURce[1]:FUNCtion:SHAPe? (returns DC)

  :SOURce[1]:MEMory:POINts? (returns 1)

  :SOURce[1]:MEMory:STARt? (returns 1)

  :SOURce[1]:SOAK? (returns 0)

:SOURce[1]:VOLTage[:LEVel]:TRIGgered:SFACtor? (returns 1.0)

:SOURce[1]:VOLTage[:LEVel]:TRIGgered:SFACtor:STATe? (returns 0)

:STATus:QUEue:DISable? (returns (+0))

:STATus:QUEue:ENABle? (returns (-440:-100,+111:+954))

:SYSTem:AZERo:CACHing:NPLCycles? (returns 0)

:SYSTem:AZERo:CACHing[:STATe]? (returns 0)

:SYSTem:AZERo:STATe? (returns 0)

:SYSTem:CCHeck? (returns 0)

:SYSTem:CCHeck:RESistance? (returns 50)

:SYSTem:GUARd? (returns CABL)

:SYSTem:KEY? (returns 0)

:SYSTem:LFRequency:AUTO? (returns 0)

:SYSTem:MEP[:STATe]? (returns 1)

:SYSTem:MEP:HOLDoff? (returns 0)

:SYSTem:RCMode? (returns SING)

:SYSTem:RWLock? (returns 0)

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:ILINe? (returns 1)

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:INPut? (returns NONE)

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:OLINe? (returns 2)

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:OUTPut? (returns NONE)

:TRIGger:SEQuence2:SOURce? (returns IMM)

:TRIGger:SEQuence2:TOUT? [DEFault|MINimum|MAXimum] (returns 0.0)

Conventional commands not supported by B2900

:ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:ILINe <1|2|3|4>

:ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:OLINe <1|2|3|4>

:ARM[:SEQuence[1]][:LAYer[1]][:TCONfigure][:ASYNchronous]:OUTPut <TENTer|TEXit|NONE>

:CALCulate2:CLIMits:FAIL:SMLocation <NR1>|NEXT

:CALCulate2:CLIMits:PASS:SMLocation <NR1>|NEXT

:DISPlay[:WINDow[1]]:DATA?

:DISPlay:WINDow2:DATA?

:OUTPut[1]:INTerlock:STATe <Bool>

[:SENSe[1]]:AVERage:STATe <Bool>

[:SENSe[1]]:CURRent[:DC]:PROTection:RSYNchronize <Bool>

[:SENSe[1]]:VOLTage[:DC]:PROTection:RSYNchronize <Bool>

:SOURce[1]:CLEar:AUTO:MODE <ALWays|TCOunt>

:SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACtor <NRf>

:SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACtor:STATe <Bool>

:SOURce[1]:FUNCtion[:MODE] <VOLTage|CURRent|MEMory>

:SOURce[1]:FUNCtion:SHAPe <DC|PULSe>

:SOURce[1]:MEMory:POINts <NR1>

:SOURce[1]:MEMory:RECall <NR1>

:SOURce[1]:MEMory:SAVE <NR1>

:SOURce[1]:MEMory:STARt <NR1>

:SOURce[1]:SOAK <NRf>

:SOURce[1]:VOLTage[:LEVel]:TRIGgered:SFACtor <NRf>

:SOURce[1]:VOLTage[:LEVel]:TRIGgered:SFACtor:STATe <Bool>

:STATus:QUEue:DISable (NR1 list)

:STATus:QUEue:ENABle (NR1 list)

:SYSTem:CCHeck ON|OFF

:SYSTem:LFRequency:AUTO <Bool>

:SYSTem:MEP[:STATe]

:SYSTem:RCMode <SINGle|MULTiple>

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:ILINe <1|2|3|4>

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:INPut <SOURce|DELay|SENSe|NONE>

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:OLINe <1|2|3|4>

:TRIGger[:SEQuence[1]][:TCONfigure][:ASYNchronous]:OUTPut <SOURce|DELay|SENSe|NONE>