

# Keysight Technologies B2900 Series Source/Measure Unit

SCPI Command  
Reference

# Notices

## Copyright Notice

© Keysight Technologies 2011-2019

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies as governed by United States and international copyright laws.

## Manual Part Number

B2910-90030

## Edition

Edition 1, March 2011

Edition 2, July 2011

Edition 3, March 2012

Edition 4, August 2013

Edition 5, October 2014

Edition 6, December 2015

Edition 7, December 2016

Edition 8, March 2019

## Published by:

Keysight Technologies Japan K.K.

9-1, Takakura-cho, Hachioji-shi, Tokyo

192-0033 Japan

## 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.

## U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at <http://www.keysight.com/find/sweula>. The

license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

## 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, KEYSIGHT 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. KEYSIGHT 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 KEYSIGHT 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.

## 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.keysight.com/find/GPLV2](http://www.keysight.com/find/GPLV2)

## Declaration of Conformity

Declarations of Conformity for this product and for other Keysight products may be downloaded from the Web. Go to [www.keysight.com/go/conformity](http://www.keysight.com/go/conformity). You can then search by product number to find the latest Declaration of Conformity.

## Latest Information

To get the latest firmware/software/electronic manuals/specifications/support information, go to [www.keysight.com](http://www.keysight.com) and type in the product number in the Search field at the top of the page.

## 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 Keysight B2900 manual feedback form](#).

We respect your privacy. Be assured that Keysight will never sell or rent your information. Nor will Keysight 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 Keysight B2900 Source/Measure Unit series over the remote interface using the SCPI programming language. The Keysight B2900 supports the SCPI programming language on all of its remote I/O interfaces.

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.

2. **“Subsystem Command Summary”**

Lists the B2900 SCPI subsystem commands and summary descriptions.

3. **“Common Commands”**

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

4. **“Subsystem Commands”**

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

5. **“Error Messages”**

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

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

## 1. Programming Basics

|                                   |      |
|-----------------------------------|------|
| SCPI Commands                     | 1-3  |
| Multiple Commands in a Message    | 1-3  |
| Moving Between Subsystems         | 1-4  |
| Including Common Commands         | 1-5  |
| Using Queries                     | 1-5  |
| Coupled Commands                  | 1-5  |
| SCPI Messages                     | 1-6  |
| Headers                           | 1-6  |
| Message Unit                      | 1-6  |
| Message Unit Separator            | 1-7  |
| Root Specifier                    | 1-7  |
| Query Indicator                   | 1-7  |
| Message Terminator                | 1-7  |
| Numeric Suffix                    | 1-8  |
| Channel List Parameter            | 1-8  |
| SCPI Command Completion           | 1-9  |
| Device Clear                      | 1-9  |
| SCPI Conventions and Data Formats | 1-10 |
| Data Output Format                | 1-12 |
| Status Data                       | 1-13 |
| GPIB Capability                   | 1-17 |
| Status Byte                       | 1-18 |
| Status System Diagram             | 1-19 |
| Non-Volatile Settings             | 1-23 |

## 2. Subsystem Command Summary

|                             |     |
|-----------------------------|-----|
| Setting Source/Measure Unit | 2-4 |
|-----------------------------|-----|

|  |      |
|--|------|
| Source Output Ranges . . . . .               | 2-24 |
| Measurement Ranges . . . . .                 | 2-27 |
| Sweep Steps . . . . .                        | 2-28 |
| Controlling Source/Measure Trigger . . . . . | 2-29 |
| LXI Trigger Events . . . . .                 | 2-37 |
| Reading Source/Measure Data . . . . .        | 2-46 |
| Using Advanced Functions . . . . .           | 2-60 |

### 3. Common Commands

|                 |      |
|-----------------|------|
| *CAL? . . . . . | 3-3  |
| *CLS . . . . .  | 3-4  |
| *ESE . . . . .  | 3-5  |
| *ESR? . . . . . | 3-6  |
| *IDN? . . . . . | 3-8  |
| *OPC . . . . .  | 3-9  |
| *RCL . . . . .  | 3-10 |
| *RST . . . . .  | 3-11 |
| *SAV . . . . .  | 3-12 |
| *SRE . . . . .  | 3-13 |
| *STB? . . . . . | 3-15 |
| *TRG . . . . .  | 3-16 |
| *TST? . . . . . | 3-17 |
| *WAI . . . . .  | 3-18 |

### 4. Subsystem Commands

|  |      |
|--|------|
| CALCulate Subsystem                            | 4-3  |
| :CALCulate:CLIMits:CLEar:AUTO                  | 4-3  |
| :CALCulate:CLIMits:CLEar:AUTO:DElay            | 4-3  |
| :CALCulate:CLIMits:CLEar[:IMMediate]           | 4-4  |
| :CALCulate:CLIMits:<FAIL PASS>:DIGital[:DATA]  | 4-4  |
| :CALCulate:CLIMits:MODE                        | 4-5  |
| :CALCulate:CLIMits:STATe                       | 4-5  |
| :CALCulate:CLIMits:STATe:ANY?                  | 4-6  |
| :CALCulate:CLIMits:UPDate                      | 4-6  |
| :CALCulate:DATA?                               | 4-7  |
| :CALCulate:DATA:LATest?                        | 4-8  |
| :CALCulate:DIGital:BIT                         | 4-8  |
| :CALCulate:DIGital:<BUSY EOT SOT>              | 4-9  |
| :CALCulate:FEED                                | 4-9  |
| :CALCulate:LIMit:COMpliance:DIGital[:DATA]     | 4-10 |
| :CALCulate:LIMit:COMpliance:FAIL               | 4-11 |
| :CALCulate:LIMit:FAIL?                         | 4-11 |
| :CALCulate:LIMit:FUNCTion                      | 4-12 |
| :CALCulate:LIMit:<LOWer UPPer>                 | 4-12 |
| :CALCulate:LIMit:<LOWer UPPer>:DIGital[:DATA]  | 4-13 |
| :CALCulate:LIMit:PASS:DIGital[:DATA]           | 4-13 |
| :CALCulate:LIMit:STATe                         | 4-14 |
| :CALCulate:MATH:DATA?                          | 4-14 |
| :CALCulate:MATH:DATA:LATest?                   | 4-15 |
| :CALCulate:MATH[:EXPRession]:CATalog?          | 4-16 |
| :CALCulate:MATH[:EXPRession][:DEFine]          | 4-16 |
| :CALCulate:MATH[:EXPRession]:DELete:ALL        | 4-19 |
| :CALCulate:MATH[:EXPRession]:DELete[:SELected] | 4-19 |
| :CALCulate:MATH[:EXPRession]:NAME              | 4-19 |
| :CALCulate:MATH:STATe                          | 4-20 |
| :CALCulate:MATH:UNITs                          | 4-21 |
| :CALCulate:OFFSet                              | 4-21 |

|  |      |
|--|------|
| :CALCulate:OFFSet:ACQuire . . . . .  | 4-22 |
| :CALCulate:OFFSet:STATe . . . . .  | 4-22 |
| DISPlay Subsystem . . . . .  | 4-23 |
| :DISPlay:CSET . . . . .  | 4-23 |
| :DISPlay:DIGits . . . . .  | 4-23 |
| :DISPlay:ENABle . . . . .  | 4-24 |
| :DISPlay:VIEW . . . . .  | 4-25 |
| :DISPlay[:WINDow]:DATA? . . . . .  | 4-25 |
| :DISPlay[:WINDow]:TEXT:DATA . . . . .                                      | 4-26 |
| :DISPlay[:WINDow]:TEXT:STATe . . . . .                                     | 4-26 |
| :DISPlay:ZOOM . . . . .  | 4-27 |
| FEtCh Subsystem . . . . .  | 4-28 |
| :FEtCh:ARRay? . . . . .  | 4-28 |
| :FEtCh:ARRay:<CURRent RESistance SOURce STATus TIME VOLTage>? . . . . .    | 4-29 |
| :FEtCh[:SCALar]? . . . . .   | 4-30 |
| :FEtCh[:SCALar]:<CURRent RESistance SOURce STATus TIME VOLTage>? . . . . . | 4-31 |
| FORMat Subsystem . . . . .   | 4-32 |
| :FORMat:BORDer . . . . .   | 4-32 |
| :FORMat[:DATA] . . . . .   | 4-32 |
| :FORMat:DIGital . . . . .  | 4-33 |
| :FORMat:ELEMents:CALCulate . . . . .                                       | 4-34 |
| :FORMat:ELEMents:SENSe . . . . .   | 4-35 |
| :FORMat:SREGister . . . . .  | 4-36 |
| HCOPy Subsystem . . . . .  | 4-37 |
| :HCOPy:SDUMp:DATA? . . . . .   | 4-37 |
| :HCOPy:SDUMp:FORMat . . . . .  | 4-37 |
| LXI Subsystem . . . . .  | 4-38 |
| :ARM:LXI:COUNT . . . . .   | 4-38 |



|   |      |
|---|------|
| :ARM:LXI:DElay . . . . .                        | 4-39 |
| :ARM:LXI:LAN[:SET]:DETection . . . . .          | 4-39 |
| :ARM:LXI:LAN[:SET]:ENABle . . . . .             | 4-40 |
| :ARM:LXI:LAN[:SET]:FILTer . . . . .             | 4-41 |
| :ARM:LXI:LAN[:SET]:IDENtifier . . . . .         | 4-41 |
| :LXI:EVENT:DOMain . . . . .                     | 4-42 |
| :LXI:EVENT:INPut:LAN:ADD . . . . .              | 4-42 |
| :LXI:EVENT:INPut:LAN:COUNT? . . . . .           | 4-42 |
| :LXI:EVENT:INPut:LAN:DISAbLe:ALL . . . . .      | 4-43 |
| :LXI:EVENT:INPut:LAN:LIST? . . . . .            | 4-43 |
| :LXI:EVENT:INPut:LAN:REMOve:ALL . . . . .       | 4-43 |
| :LXI:EVENT:INPut:LAN:REMOve . . . . .           | 4-44 |
| :LXI:EVENT:INPut:LAN[:SET]:CONFigure . . . . .  | 4-44 |
| :LXI:EVENT:INPut:LAN[:SET]:DElay . . . . .      | 4-45 |
| :LXI:EVENT:INPut:LAN[:SET]:DETection. . . . .   | 4-45 |
| :LXI:EVENT:INPut:LAN[:SET]:ENABle . . . . .     | 4-46 |
| :LXI:EVENT:INPut:LAN[:SET]:FILTer . . . . .     | 4-47 |
| :LXI:EVENT:INPut:LAN[:SET]:IDENtifier . . . . . | 4-47 |
| :LXI:EVENT:LOG:ALL? . . . . .                   | 4-48 |
| :LXI:EVENT:LOG:CIRCUlar[:ENABle]. . . . .       | 4-48 |
| :LXI:EVENT:LOG:CIRCUlar:FBEntry . . . . .       | 4-49 |
| :LXI:EVENT:LOG:CLEAr . . . . .                  | 4-49 |
| :LXI:EVENT:LOG:COUNT? . . . . .                 | 4-49 |
| :LXI:EVENT:LOG:ENABle . . . . .                 | 4-49 |
| :LXI:EVENT:LOG:ENTRY? . . . . .                 | 4-50 |
| :LXI:EVENT:LOG:SIZE . . . . .                   | 4-51 |
| :LXI:EVENT[:OUTPut]:LAN:ADD . . . . .           | 4-51 |
| :LXI:EVENT[:OUTPut]:LAN:COUNT? . . . . .        | 4-52 |
| :LXI:EVENT[:OUTPut]:LAN:DISAbLe:ALL . . . . .   | 4-52 |
| :LXI:EVENT[:OUTPut]:LAN:LIST? . . . . .         | 4-52 |
| :LXI:EVENT[:OUTPut]:LAN:REMOve:ALL . . . . .    | 4-52 |
| :LXI:EVENT[:OUTPut]:LAN:REMOve . . . . .        | 4-53 |

|   |      |
|---|------|
| :LXI:EVENT[:OUTPut]:LAN:SEND . . . . .              | 4-53 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:CONFigure . . . . .   | 4-53 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:DESTination . . . . . | 4-54 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:DRIVE . . . . .       | 4-55 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:ENABle . . . . .      | 4-56 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:IDENtifier . . . . .  | 4-56 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:SLOPe . . . . .       | 4-57 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:SOURce . . . . .      | 4-57 |
| :LXI:EVENT[:OUTPut]:LAN[:SET]:TSDelta . . . . .     | 4-58 |
| :LXI:IDENtify[:STATe] . . . . .                     | 4-59 |
| :LXI:MDNS:ENABle . . . . .                          | 4-59 |
| :LXI:MDNS:HNAME[:RESolved]? . . . . .               | 4-60 |
| :LXI:MDNS:SNAME:DESired . . . . .                   | 4-60 |
| :LXI:MDNS:SNAME[:RESolved]? . . . . .               | 4-60 |
| :TRIGger:LXI:LAN[:SET]:DELay . . . . .              | 4-61 |
| :TRIGger:LXI:LAN[:SET]:DETection . . . . .          | 4-61 |
| :TRIGger:LXI:LAN[:SET]:ENABle . . . . .             | 4-62 |
| :TRIGger:LXI:LAN[:SET]:FILTer . . . . .             | 4-62 |
| :TRIGger:LXI:LAN[:SET]:IDENtifier . . . . .         | 4-63 |
| MEASure Subsystem . . . . .                         | 4-64 |
| :MEASure? . . . . .                                 | 4-64 |
| :MEASure:<CURRent RESistance VOLTage>? . . . . .    | 4-65 |
| MMEMory Subsystem . . . . .                         | 4-66 |
| :MMEMory:CATalog? . . . . .                         | 4-66 |
| :MMEMory:CDIRectory . . . . .                       | 4-67 |
| :MMEMory:COPI . . . . .                             | 4-67 |
| :MMEMory:DELeTe . . . . .                           | 4-68 |
| :MMEMory:LOAD:MACRo . . . . .                       | 4-68 |
| :MMEMory:LOAD:STATe . . . . .                       | 4-68 |
| :MMEMory:MDIRectory . . . . .                       | 4-69 |
| :MMEMory:MOVE . . . . .                             | 4-69 |

|  |      |
|--|------|
| :MMEMory:RDIRECTory . . . . .                            | 4-70 |
| :MMEMory:STORE:DATA<:LIMit[:MATH]:SENSe[:ALL]> . . . . . | 4-70 |
| :MMEMory:STORE:MACRO . . . . .                           | 4-71 |
| :MMEMory:STORE:STATE . . . . .                           | 4-71 |
| :MMEMory:STORE:TRACe . . . . .                           | 4-71 |
| OUTPut Subsystem . . . . .                               | 4-73 |
| :OUTPut:FILTer:AUTO . . . . .                            | 4-73 |
| :OUTPut:FILTer[:LPASs]:FREQuency . . . . .               | 4-73 |
| :OUTPut:FILTer[:LPASs][:STATE]. . . . .                  | 4-74 |
| :OUTPut:FILTer[:LPASs]:TCOnstant . . . . .               | 4-75 |
| :OUTPut:HCAPacitance[:STATE] . . . . .                   | 4-75 |
| :OUTPut:LOW . . . . .                                    | 4-76 |
| :OUTPut:OFF:AUTO . . . . .                               | 4-76 |
| :OUTPut:OFF:MODE . . . . .                               | 4-77 |
| :OUTPut:ON:AUTO . . . . .                                | 4-78 |
| :OUTPut:PROTection[:STATE]. . . . .                      | 4-79 |
| :OUTPut:RECall . . . . .                                 | 4-79 |
| :OUTPut:SAVE . . . . .                                   | 4-79 |
| :OUTPut[:STATE]. . . . .                                 | 4-80 |
| PROGram Subsystem . . . . .                              | 4-81 |
| :PROGram:CATalog? . . . . .                              | 4-81 |
| :PROGram:PON:COpy . . . . .                              | 4-81 |
| :PROGram:PON:DELeTe . . . . .                            | 4-81 |
| :PROGram:PON:RUm . . . . .                               | 4-82 |
| :PROGram[:SELeCted]:APPend . . . . .                     | 4-82 |
| :PROGram[:SELeCted]:DEFine . . . . .                     | 4-83 |
| :PROGram[:SELeCted]:DELeTe:ALL . . . . .                 | 4-84 |
| :PROGram[:SELeCted]:DELeTe[:SELeCted] . . . . .          | 4-84 |
| :PROGram[:SELeCted]:EXECute . . . . .                    | 4-84 |
| :PROGram[:SELeCted]:NAME . . . . .                       | 4-85 |
| :PROGram[:SELeCted]:STATE . . . . .                      | 4-85 |

|   |       |
|---|-------|
| :PROGram[:SELEcted]:WAIT?                                       | 4-86  |
| :PROGram:VARiable   | 4-87  |
| READ Subsystem  | 4-88  |
| :READ:ARRay?  | 4-88  |
| :READ:ARRay:<CURRent RESistance SOURce STATus TIME VOLTage>?    | 4-89  |
| :READ[:SCALar]?   | 4-90  |
| :READ[:SCALar]:<CURRent RESistance SOURce STATus TIME VOLTage>? | 4-91  |
| SENSe Subsystem   | 4-93  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:APERture          | 4-93  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:APERture:AUTO     | 4-94  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:NPLCycles         | 4-94  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:NPLCycles:AUTO    | 4-95  |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:PROTection[:LEVel][:BOTH]    | 4-96  |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:PROTection[:LEVel]:NEGative  | 4-97  |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:PROTection[:LEVel]:POSitive  | 4-97  |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:PROTection:TRIPped?          | 4-98  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:RANGe:AUTO        | 4-99  |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:RANGe:AUTO:LLIMit | 4-99  |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:RANGe:AUTO:MODE              | 4-100 |
| :SENSe:<CURRent[:DC] VOLTage[:DC]>:RANGe:AUTO:THReshold         | 4-101 |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:RANGe:AUTO:ULIMit | 4-102 |
| :SENSe:<CURRent[:DC] RESistance VOLTage[:DC]>:RANGe[:UPPer]     | 4-103 |
| :SENSe:DATA?  | 4-104 |
| :SENSe:DATA:LATest?   | 4-105 |
| :SENSe:FUNCTion:OFF   | 4-105 |
| :SENSe:FUNCTion:OFF:ALL   | 4-106 |
| :SENSe:FUNCTion:OFF:COUNT?                                      | 4-106 |
| :SENSe:FUNCTion[:ON]  | 4-106 |

|   |       |
|---|-------|
| :SENSe:FUNctIon[:ON]:ALL . . . . .                                    | 4-107 |
| :SENSe:FUNctIon[:ON]:COUNT? . . . . .                                 | 4-107 |
| :SENSe:FUNctIon:STATe? . . . . .                                      | 4-107 |
| :SENSe:REMOte . . . . .   | 4-108 |
| :SENSe:RESistance:MODE . . . . .                                      | 4-108 |
| :SENSe:RESistance:OCOMpensated . . . . .                              | 4-109 |
| :SENSe:TOUTput:SIGNal . . . . .                                       | 4-110 |
| :SENSe:TOUTput[:STATe]. . . . .                                       | 4-110 |
| :SENSe:WAIT:AUTO . . . . .  | 4-111 |
| :SENSe:WAIT:GAIN . . . . .  | 4-111 |
| :SENSe:WAIT:OFFSet . . . . .  | 4-112 |
| :SENSe:WAIT[:STATe]. . . . .  | 4-112 |
| SOURce Subsystem . . . . .  | 4-114 |
| [:SOURce]:<CURRent VOLTage>:<CENTer SPAN> . . . . .                   | 4-114 |
| [:SOURce]:<CURRent VOLTage>[:LEVel][:IMMediate][:AMPLitude] . . . . . | 4-114 |
| [:SOURce]:<CURRent VOLTage>[:LEVel]:TRIGgered[:AMPLitude] . . . . .   | 4-115 |
| [:SOURce]:<CURRent VOLTage>:MODE . . . . .                            | 4-116 |
| [:SOURce]:<CURRent VOLTage>:POINts . . . . .                          | 4-116 |
| [:SOURce]:<CURRent VOLTage>:RANGe . . . . .                           | 4-117 |
| [:SOURce]:<CURRent VOLTage>:RANGe:AUTO . . . . .                      | 4-118 |
| [:SOURce]:<CURRent VOLTage>:RANGe:AUTO:LLIMit . . . . .               | 4-118 |
| [:SOURce]:<CURRent VOLTage>:RANGe:RPPriority . . . . .                | 4-119 |
| [:SOURce]:<CURRent VOLTage>:<START STOP>. . . . .                     | 4-120 |
| [:SOURce]:<CURRent VOLTage>:STEP . . . . .                            | 4-120 |
| [:SOURce]:<CURRent VOLTage>:TRANSient:SPEed . . . . .                 | 4-121 |
| [:SOURce]:DIGital:DATA . . . . .                                      | 4-122 |
| [:SOURce]:DIGital:EXTernal:FUNCTion . . . . .                         | 4-122 |
| [:SOURce]:DIGital:EXTernal:POLarity . . . . .                         | 4-123 |
| [:SOURce]:DIGital:EXTernal:TOUTput[:EDGE]:POSition . . . . .          | 4-124 |
| [:SOURce]:DIGital:EXTernal:TOUTput[:EDGE]:WIDTH . . . . .             | 4-124 |
| [:SOURce]:DIGital:EXTernal:TOUTput:TYPE . . . . .                     | 4-125 |

|  |       |
|--|-------|
| [:SOURce]:DIGital:INTernal:TOUTput[:EDGE]:POSition . . . . .                                 | 4-125 |
| [:SOURce]:FUNCTion:MODE . . . . .  | 4-126 |
| [:SOURce]:FUNCTion[:SHAPE] . . . . .   | 4-126 |
| [:SOURce]:FUNCTion:TRIGgered:CONTinuous . . . . .  | 4-127 |
| [:SOURce]:LIST:<CURRent VOLTage> . . . . .   | 4-128 |
| [:SOURce]:LIST:<CURRent VOLTage>:APPend . . . . .  | 4-128 |
| [:SOURce]:LIST:<CURRent VOLTage>:POINts? . . . . .   | 4-129 |
| [:SOURce]:LIST:<CURRent VOLTage>:STARt . . . . .   | 4-129 |
| [:SOURce]:PULSe:DELay . . . . .  | 4-130 |
| [:SOURce]:PULSe:WIDTh . . . . .  | 4-130 |
| [:SOURce]:SWEep:DIRection . . . . .  | 4-131 |
| [:SOURce]:SWEep:POINts . . . . .   | 4-131 |
| [:SOURce]:SWEep:RANGing . . . . .  | 4-132 |
| [:SOURce]:SWEep:SPACing . . . . .  | 4-133 |
| [:SOURce]:SWEep:STAir . . . . .  | 4-134 |
| [:SOURce]:TOUTput:SIGNal . . . . .   | 4-134 |
| [:SOURce]:TOUTput[:STATe] . . . . .  | 4-135 |
| [:SOURce]:WAIT:AUTO . . . . .  | 4-135 |
| [:SOURce]:WAIT:GAIN . . . . .  | 4-136 |
| [:SOURce]:WAIT:OFFSet . . . . .  | 4-136 |
| [:SOURce]:WAIT[:STATe] . . . . .   | 4-137 |
| STATus Subsystem . . . . .   | 4-138 |
| :STATus:<MEASurement OPERation QUESTionable>:CONDition? . . . . .                            | 4-138 |
| :STATus:<MEASurement OPERation QUESTionable>:ENABLE . . . . .                                | 4-141 |
| :STATus:<MEASurement OPERation QUESTionable>[:EVENT]? . . . . .                              | 4-141 |
| :STATus:<MEASurement OPERation QUESTionable>:NTRansition . . . . .                           | 4-142 |
| :STATus:<MEASurement OPERation QUESTionable>:PTRansition . . . . .                           | 4-143 |
| :STATus:PRESet . . . . .   | 4-143 |
| :STATus:QUESTionable:<CALibration CURRent TEMPerature TEST VOLTage><br>:CONDition? . . . . . | 4-144 |
| :STATus:QUESTionable:<CALibration CURRent TEMPerature TEST VOLTage><br>:ENABLE . . . . .     | 4-146 |

|  |       |
|--|-------|
| :STATus:QUEStionable:<CALibration CURRent TEMPerature TEST VOLTage><br>[:EVENT]? . . . . .   | 4-146 |
| :STATus:QUEStionable:<CALibration CURRent TEMPerature TEST VOLTage><br>:NTRansition. . . . . | 4-147 |
| :STATus:QUEStionable:<CALibration CURRent TEMPerature TEST VOLTage><br>:PTRansition. . . . . | 4-148 |
| SYSTem Subsystem . . . . .   | 4-149 |
| :SYSTem:BEEPer[:IMMEDIATE] . . . . .   | 4-149 |
| :SYSTem:BEEPer:STATe . . . . .   | 4-149 |
| :SYSTem:COMMunicate:ENABle . . . . .   | 4-150 |
| :SYSTem:COMMunicate:GPIB[:SELF]:ADDReSS . . . . .  | 4-150 |
| :SYSTem:COMMunicate:LAN:ADDReSS . . . . .  | 4-151 |
| :SYSTem:COMMunicate:LAN:BStatus?. . . . .  | 4-151 |
| :SYSTem:COMMunicate:<LAN TCPIp>:CONTRol? . . . . .   | 4-152 |
| :SYSTem:COMMunicate:LAN:DHCP . . . . .   | 4-152 |
| :SYSTem:COMMunicate:LAN:DNS . . . . .  | 4-153 |
| :SYSTem:COMMunicate:LAN:DOMain?. . . . .   | 4-153 |
| :SYSTem:COMMunicate:LAN:<GATE GATeway> . . . . .   | 4-153 |
| :SYSTem:COMMunicate:LAN:<HNAME HOSTname> . . . . .   | 4-154 |
| :SYSTem:COMMunicate:LAN:MAC? . . . . .   | 4-154 |
| :SYSTem:COMMunicate:LAN:SMASk . . . . .  | 4-155 |
| :SYSTem:COMMunicate:LAN:TELNet:PROMpt. . . . .   | 4-155 |
| :SYSTem:COMMunicate:LAN:TELNet:WMESsage . . . . .  | 4-156 |
| :SYSTem:COMMunicate:LAN:UPDate . . . . .   | 4-156 |
| :SYSTem:COMMunicate:LAN:WINS . . . . .   | 4-157 |
| :SYSTem:DATA:QUANtity? . . . . .   | 4-157 |
| :SYSTem:DATE . . . . .   | 4-158 |
| :SYSTem:ERRor:ALL? . . . . .   | 4-158 |
| :SYSTem:ERRor:CODE:ALL? . . . . .  | 4-158 |
| :SYSTem:ERRor:CODE[:NEXT]? . . . . .   | 4-159 |
| :SYSTem:ERRor:COUNt? . . . . .   | 4-159 |
| :SYSTem:ERRor[:NEXT]? . . . . .  | 4-159 |

|  |       |
|--|-------|
| :SYSTem:FAN:MODE . . . . .                           | 4-160 |
| :SYSTem:GROup[:DEFine]. . . . .                      | 4-160 |
| :SYSTem:GROup:RESet . . . . .                        | 4-161 |
| :SYSTem:INTerlock:TRIPped?. . . . .                  | 4-161 |
| :SYSTem:LANGuage . . . . .                           | 4-162 |
| :SYSTem:LFRequency . . . . .                         | 4-162 |
| :SYSTem:LOCK:NAME?. . . . .                          | 4-163 |
| :SYSTem:LOCK:OWNer?. . . . .                         | 4-163 |
| :SYSTem:LOCK:RELease . . . . .                       | 4-164 |
| :SYSTem:LOCK:REQuest?. . . . .                       | 4-164 |
| :SYSTem:PERSon[:MANufacturer] . . . . .              | 4-165 |
| :SYSTem:PERSon[:MANufacturer]:DEFault . . . . .      | 4-166 |
| :SYSTem:PERSon:MODEl . . . . .                       | 4-166 |
| :SYSTem:PERSon:MODEl:DEFault . . . . .               | 4-166 |
| :SYSTem:PON . . . . .                                | 4-167 |
| :SYSTem:PRESet . . . . .                             | 4-167 |
| :SYSTem:SET . . . . .                                | 4-167 |
| :SYSTem:TIME . . . . .                               | 4-168 |
| :SYSTem:TIME:TIMer:COUNT? . . . . .                  | 4-168 |
| :SYSTem:TIME:TIMer:COUNT:RESet:AUTO. . . . .         | 4-168 |
| :SYSTem:TIME:TIMer:COUNT:RESet[:IMMediate] . . . . . | 4-169 |
| :SYSTem:VERSion? . . . . .                           | 4-169 |
| TRACe Subsystem . . . . .                            | 4-170 |
| :TRACe:CLEar . . . . .                               | 4-170 |
| :TRACe:DATA? . . . . .                               | 4-170 |
| :TRACe:FEED . . . . .                                | 4-171 |
| :TRACe:FEED:CONTRol . . . . .                        | 4-171 |
| :TRACe:FREE? . . . . .                               | 4-172 |
| :TRACe:POINts . . . . .                              | 4-172 |
| :TRACe:POINts:ACTual? . . . . .                      | 4-173 |
| :TRACe:STATistic:DATA? . . . . .                     | 4-173 |



|   |       |
|---|-------|
| :TRACe:STATistic:FORMat . . . . .                                 | 4-173 |
| :TRACe:TSTamp:FORMat . . . . .                                    | 4-174 |
| TRIGGer Subsystem. . . . .  | 4-175 |
| :ABORt<:ACQuire :TRANsient[:ALL]> . . . . .                       | 4-175 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:IMMediate]. . . . .              | 4-175 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:BYPass . . . . .          | 4-176 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:COUNt . . . . .           | 4-176 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:DELay . . . . .           | 4-177 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:SOURce:LAN . . . . .      | 4-178 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:SOURce[:SIGNal] . . . . . | 4-178 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:TIMer. . . . .            | 4-179 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:TOUTput:SIGNal. . . . .   | 4-180 |
| :ARM<:ACQuire :TRANsient[:ALL]>[:LAYer]:TOUTput[:STATe] . . . . . | 4-181 |
| :IDLE<:ACQuire :TRANsient[:ALL]>? . . . . .                       | 4-181 |
| :INITiate[:IMMediate]<:ACQuire :TRANsient[:ALL]> . . . . .        | 4-182 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:BYPass. . . . .               | 4-182 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:COUNt. . . . .                | 4-183 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:DELay . . . . .               | 4-183 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>[:IMMediate]. . . . .          | 4-184 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:SOURce:LAN . . . . .          | 4-184 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:SOURce[:SIGNal] . . . . .     | 4-185 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:TIMer . . . . .               | 4-186 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:TOUTput:SIGNal . . . . .      | 4-187 |
| :TRIGGer<:ACQuire :TRANsient[:ALL]>:TOUTput[:STATe]. . . . .      | 4-187 |

## 5. Error Messages

|                                  |      |
|----------------------------------|------|
| No Error . . . . .               | 5-3  |
| Command Error . . . . .          | 5-4  |
| Execution Error . . . . .        | 5-8  |
| Device-Dependent Error . . . . . | 5-10 |

Query Error ..... 5-11

B2900 Specific Error..... 5-12

6. Using Your Existing Programs

Conventional commands supported by B2900 ..... 6-3

Conventional commands partially supported by B2900..... 6-14

Conventional commands not supported by B2900..... 6-17

---

**1**

# **Programming Basics**

This chapter describes a basic information for programming Keysight 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>
```

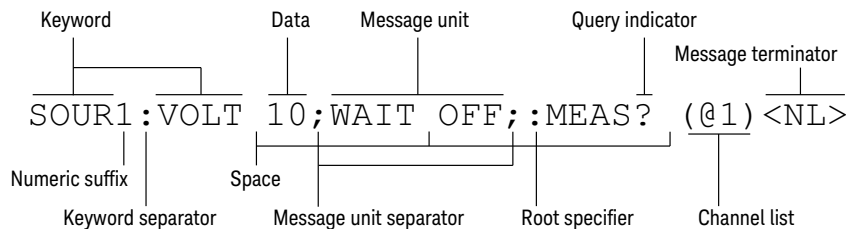
will always be correct because the commands are not executed until the message terminator is received. Because the range and setting information is received as a set, no range/setting conflict occurs.

## 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 signal lines, etc. The numeric suffix is an optional character, and can be expressed by  $[c]$ ,  $[d]$ ,  $[h]$ ,  $[i]$ ,  $[j]$ ,  $[m]$ , 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 (1) or lower half (2), respectively, of the display area on the front panel display.

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

$i$  is the integer 1 or 2, used to specify the internal trigger line 1 or 2, respectively.

$j$  is the integer 1 or 2, used to specify the primary (1) or secondary (2), respectively.

$m$  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 must be placed there.  |
| 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.<br><br>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), #B1101 (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.<br><br>Channel list: Parenthetical data beginning with “@”<br><br>Group list: Parenthetical data beginning with “@”<br><br>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](#). This table shows the meaningful bits only. Ignore the other bits.

**Table 1-2**

**Bit Definitions of Status Data**

| Bit      | Description   | Decimal value   |
|----------|---|---|
| 0        | 0: Voltage source<br>1: Current source  | 0 or 1  |
| 1 and 2  | Compliance condition<br>0: No or 1, 2, 3: Yes   | 0, 2, 4, or 6   |
| 3        | Over voltage condition<br>0: No or 1: Yes   | 0 or 8  |
| 4        | Over current condition<br>0: No or 1: Yes   | 0 or 16   |
| 5        | High temperature condition<br>0: No or 1: Yes   | 0 or 32   |
| 13       | Measurement range overflow<br>0: No or 1: Yes   | 0 or 8192   |
| 14       | Offset compensation enable condition<br>0: No or 1: Yes   | 0 or 16384  |
| 16 to 20 | Composite limit test result, 0 to 31<br>See <a href="#">Table 1-3</a> for the Sorting mode test result and<br><a href="#">Table 1-4</a> for the Grading mode test result. | 0 or<br>(1 to 31)×2 <sup>16</sup><br>(0 or 65536 to<br>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<br>Serial Poll: YES<br>Talk Only Mode: NO<br>Unaddress if MLA (my listen address): YES |
| Listener            | L4   | Basic Listener: YES<br>Unaddress if MTA (my talk address): YES<br>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**

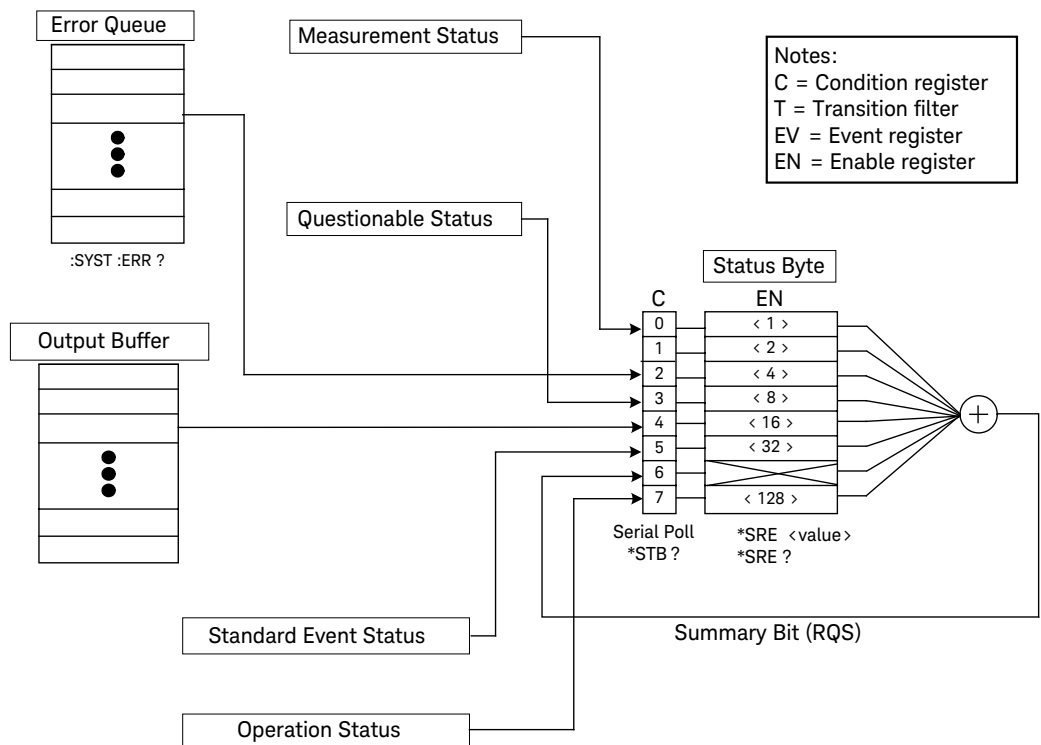
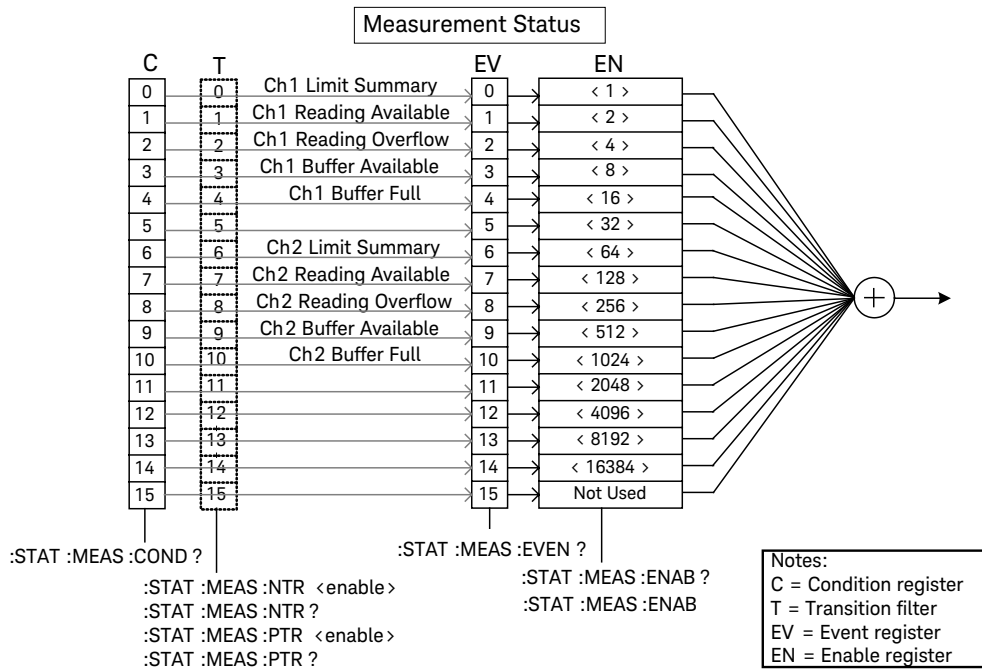


Figure 1-2

Measurement Status register



**Figure 1-3 Questionable Status register**

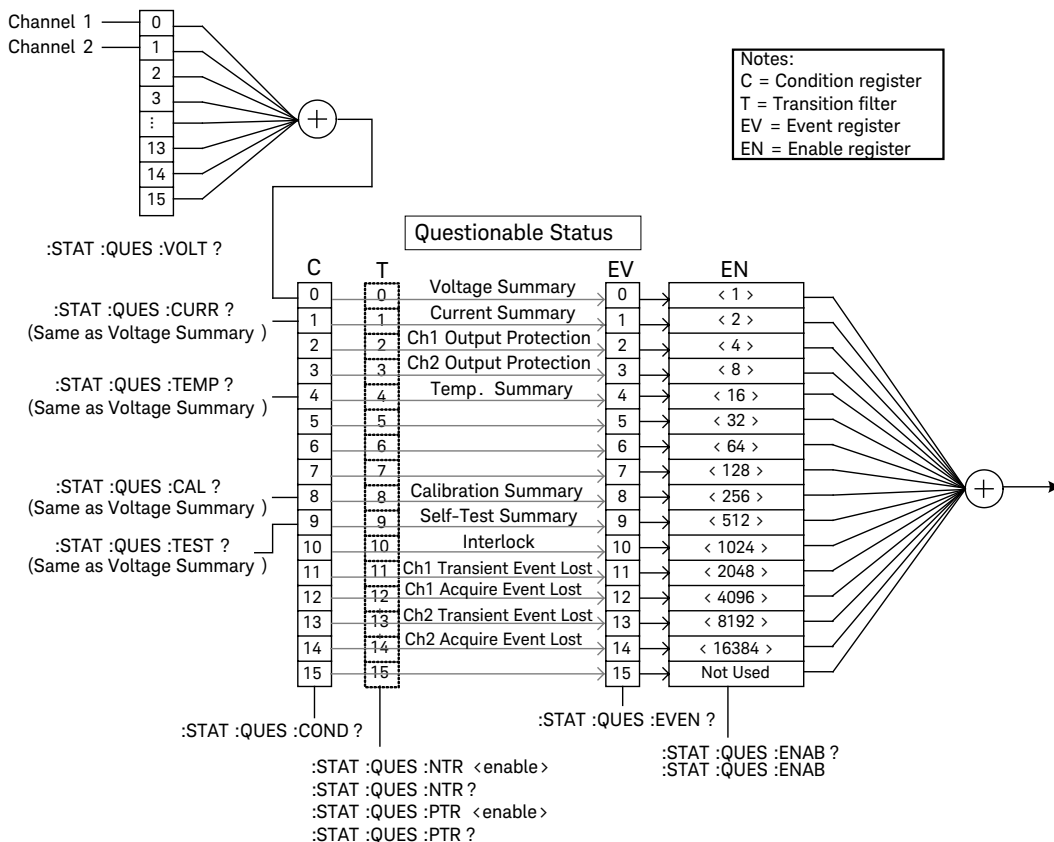


Figure 1-4

### Standard Event Status register

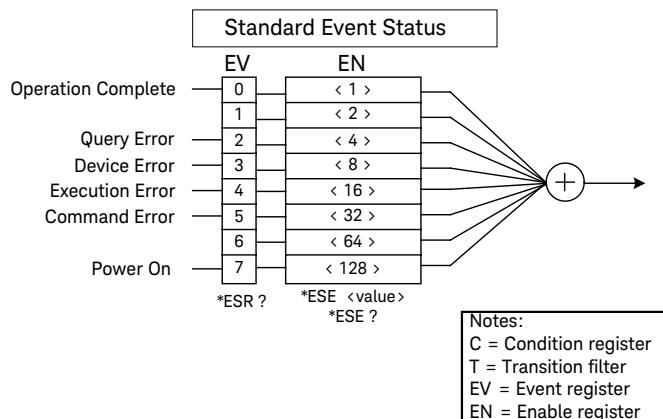
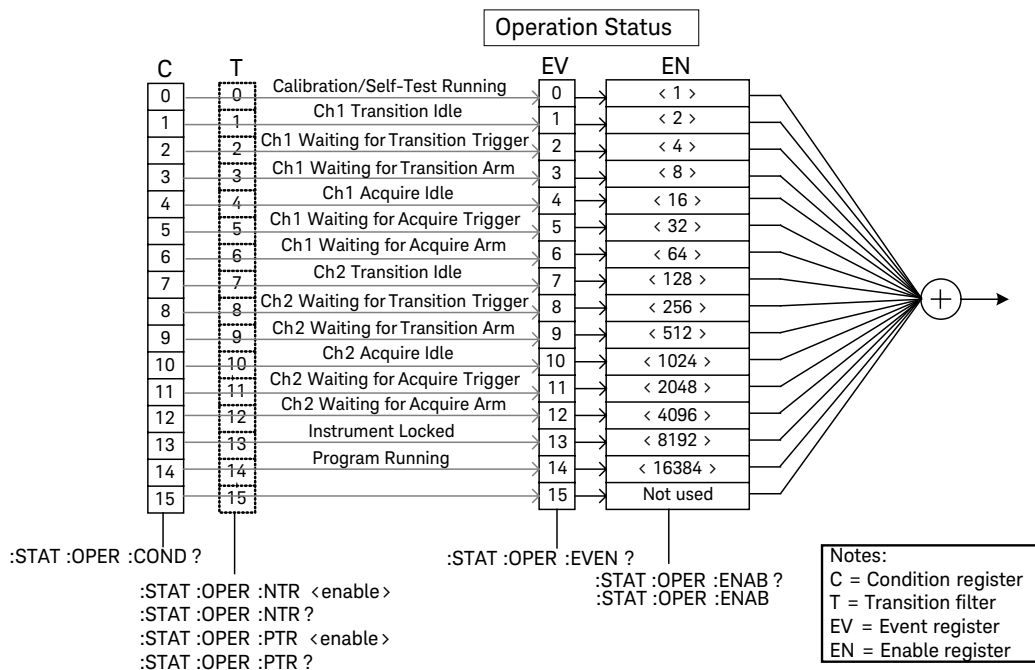


Figure 1-5

### 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                    | K-B29xxA- <i>nnnnn</i><br>B29xxA: model number<br><i>nnnnn</i> : 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   |
| GPIO address                | 23  |
| LXI identify                | Disabled  |
| GPIO 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 Keysight B2900A Series |

**Table 1-6**

**Other Non-volatile Settings**

| Setup item                   | Factory default setting                      |
|------------------------------|--|
| Channel grouping             | “1” for 1-ch models<br>“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                                       |
| V/I Limit for each polarity  | OFF  |
| Immediate V/I update by knob | OFF  |



This chapter lists all of the SCPI subsystem commands for Keysight 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
  - “LXI Subsystem,” for using LXI trigger event functions
- “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**

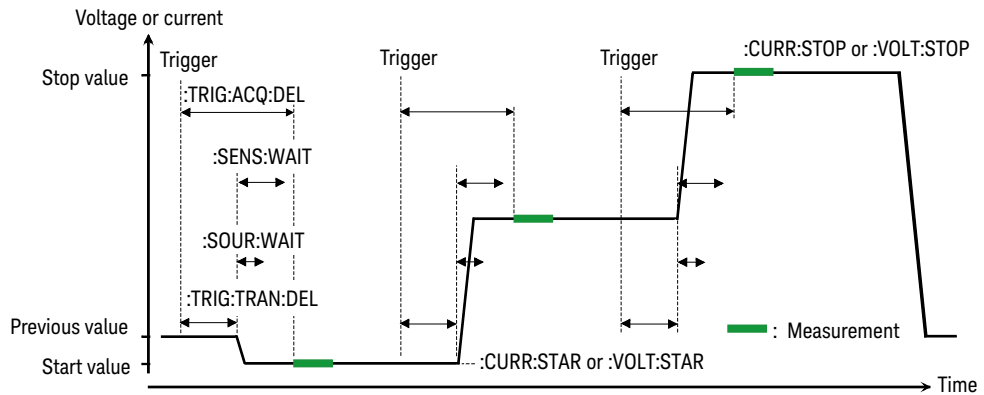
For details on numeric suffixes  $[c]$ ,  $[d]$ ,  $[h]$ ,  $[i]$ ,  $[j]$ ,  $[m]$ , and  $[n]$ , see [“Numeric Suffix” on page 1-8](#).

---

## Setting Source/Measure Unit

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

Staircase sweep source :FUNC DC, :CURR:MODE SWE or :VOLT:MODE SWE



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

Pulsed sweep source :FUNC PULS, :CURR:MODE SWE or :VOLT:MODE SWE

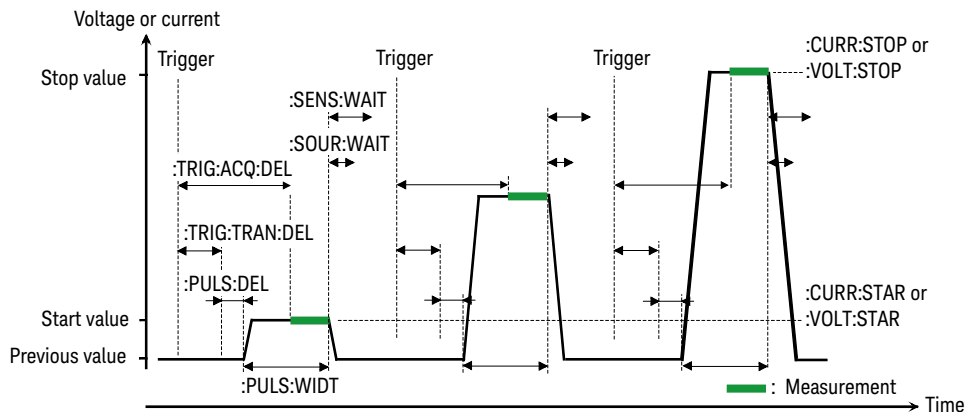


Figure 2-3

Variety of Sweep Outputs

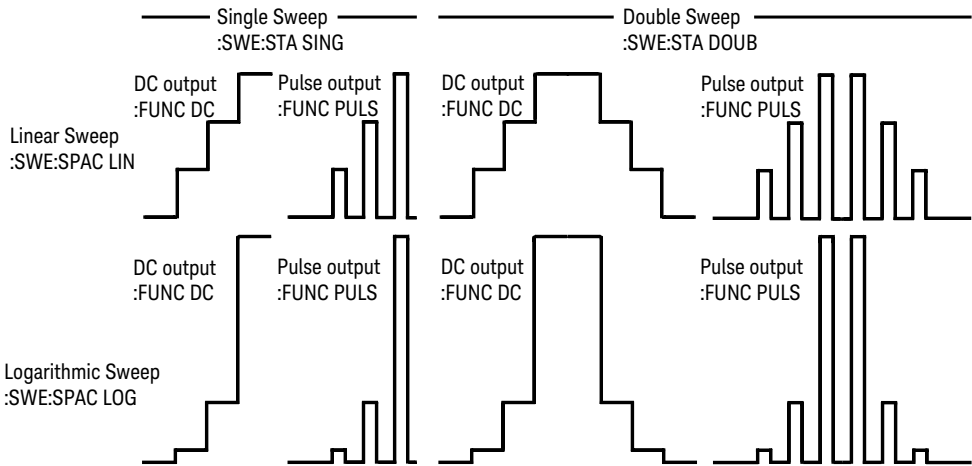
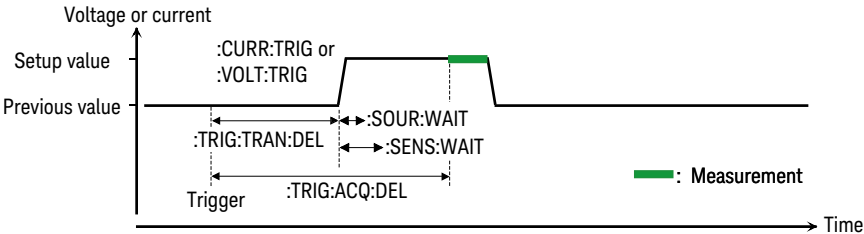


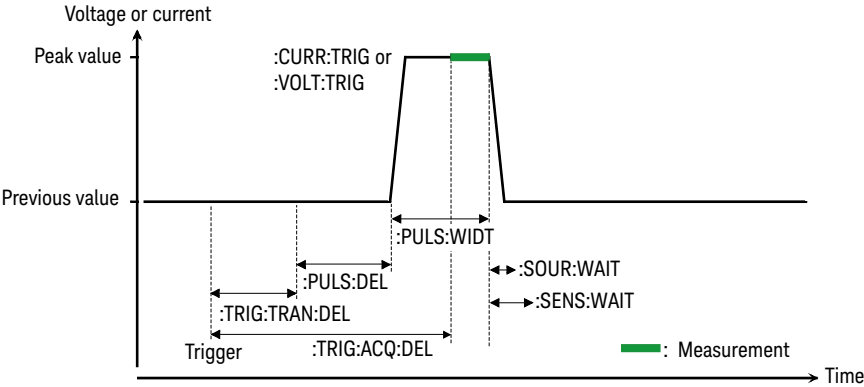
Figure 2-4

To Perform Spot Measurement

Constant source :FUNC DC, :CURR:MODE FIX or :VOLT:MODE FIX



Pulse source :FUNC PULS, :CURR:MODE FIX or :VOLT:MODE FIX



Subsystem Command Summary  
Setting Source/Measure Unit

**Table 2-1**                      **SOURce Subsystem**

| Command  | Summary   | Reset setting |
|--|---|---------------|
| [:SOUR [c]] :CURR:CENT <i>data</i><br>[:SOUR [c]] :VOLT:CENT <i>data</i><br>[:SOUR [c]] :CURR:CENT? [DEFault MINimum MAXimum]<br>[:SOUR [c]] :VOLT:CENT? [DEFault MINimum MAXimum]   | Sets the center or span value of the current or voltage sweep output.<br><br><i>data</i> =MINimum MAXimum DEFault minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.  | 0             |
| [:SOUR [c]] :CURR:SPAN <i>data</i><br>[:SOUR [c]] :VOLT:SPAN <i>data</i><br>[:SOUR [c]] :CURR:SPAN? [DEFault MINimum MAXimum]<br>[:SOUR [c]] :VOLT:SPAN? [DEFault MINimum MAXimum]   | 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.<br><br>$center = (start + stop)/2$<br>$span = stop - start$            | 0             |
| [:SOUR [c]] :CURR[:LEV] [:IMM] [:AMPL] <i>level</i><br>[:SOUR [c]] :VOLT[:LEV] [:IMM] [:AMPL] <i>level</i><br>[:SOUR [c]] :CURR[:LEV] [:IMM] [:AMPL]? [DEFault MINimum MAXimum]<br>[:SOUR [c]] :VOLT[:LEV] [:IMM] [:AMPL]? [DEFault MINimum MAXimum] | Changes the output level of the specified source channel immediately.<br><br><i>level</i> =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] <i>level</i><br>[:SOUR [c]] :VOLT[:LEV] :TRIG[:AMPL] <i>level</i><br>[:SOUR [c]] :CURR[:LEV] :TRIG[:AMPL]? [DEFault MINimum MAXimum]<br>[:SOUR [c]] :VOLT[:LEV] :TRIG[:AMPL]? [DEFault MINimum MAXimum]         | Changes the output level of the specified source channel immediately by receiving a trigger from the trigger source set by the :TRIG[c]<:ACQ :TRAN[:ALL]>:SOUR command.<br><br><i>level</i> =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 <i>mode</i><br>[:SOUR [c]] :VOLT:MODE <i>mode</i><br>[:SOUR [c]] :CURR:MODE?<br>[:SOUR [c]] :VOLT:MODE?   | Selects the source mode, fixed, list sweep, or sweep, of the specified source channel.<br><br><i>mode</i> =SWEep LIST FIXed   | FIX  |
| [:SOUR [c]] :CURR:POIN <i>points</i><br>[:SOUR [c]] :VOLT:POIN <i>points</i><br>[:SOUR [c]] :CURR:POIN? [DEFault M INimum MAXimum]<br>[:SOUR [c]] :VOLT:POIN? [DEFault M INimum MAXimum] | Sets the number of sweep steps for the current or voltage sweep output.<br><br><i>points</i> =MINimum MAXimum DEFault or integer (see “Sweep Steps” on page 2-28)<br><br>The points value can be expressed by the following formula, using the step value set by the<br>[:SOUR[c]]:<CURR VOLT>:STEP command and the span value set by the<br>[:SOUR[c]]:<CURR VOLT>:SPAN command.<br><br>$points = span/step + 1$ (where <i>step</i> is not 0)<br><br><i>points</i> =1 sets <i>step</i> =0. | 1  |
| [:SOUR [c]] :CURR:RANG <i>range</i><br>[:SOUR [c]] :VOLT:RANG <i>range</i><br>[:SOUR [c]] :CURR:RANG?<br>[: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.<br><br><i>range</i> =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 <i>mode</i><br>[:SOUR [c]] :VOLT:RANG:AUTO <i>mode</i><br>[:SOUR [c]] :CURR:RANG:AUTO?<br>[:SOUR [c]] :VOLT:RANG:AUTO?                                       | Enables or disables the automatic ranging function for the specified source channel.<br><br><i>mode</i> =1 ON 0 OFF   | ON   |

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary  | Reset setting  |
|--|--|--|
| [:SOUR [c]] :CURR:RANG:AUTO:LLIM <i>range</i><br>[:SOUR [c]] :VOLT:RANG:AUTO:LLIM <i>range</i><br>[:SOUR [c]] :CURR:RANG:AUTO:LLIM? [DEFault   MINimum   MAXimum]<br>[: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.<br><br><i>range</i> =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:RANG:RPR <i>mode</i><br>[:SOUR [c]] :VOLT:RANG:RPR <i>mode</i><br>[:SOUR [c]] :CURR:RANG:RPR?<br>[:SOUR [c]] :VOLT:RANG:RPR?   | Selects the output pulse priority mode, low noise priority or transient speed priority, of the specified source channel.<br><br><i>mode</i> =NOISe TRANsient   | NOIS   |
| [:SOUR [c]] :CURR:STAR <i>data</i><br>[:SOUR [c]] :VOLT:STAR <i>data</i><br>[:SOUR [c]] :CURR:STAR? [DEFault   MINimum   MAXimum]<br>[:SOUR [c]] :VOLT:STAR? [DEFault   MINimum   MAXimum]   | Sets the start or stop value for the current or voltage sweep output.<br><br><i>data</i> =MINimum MAXimum DEFault  minimum to maximum source value, in A or V. See “Source Output Ranges” on page 2-24.  | 0  |
| [:SOUR [c]] :CURR:STOP <i>data</i><br>[:SOUR [c]] :VOLT:STOP <i>data</i><br>[:SOUR [c]] :CURR:STOP? [DEFault   MINimum   MAXimum]<br>[:SOUR [c]] :VOLT:STOP? [DEFault   MINimum   MAXimum]   | 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.<br><br>$start = center - span/2$<br>$stop = center + span/2$      | 0  |
| [:SOUR [c]] :CURR:TRAN:SPE <i>mode</i><br>[:SOUR [c]] :VOLT:TRAN:SPE <i>mode</i><br>[:SOUR [c]] :CURR:TRAN:SPE?<br>[:SOUR [c]] :VOLT:TRAN:SPE?   | Selects the transient speed mode, normal or fast, of the specified source channel.<br><br><i>mode</i> =NORMal FAST   | NORM   |

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

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary   | Reset setting    |
|--|---|------------------|
| [:SOUR]:DIG:EXT[n]:TOUT[:EDGE]:WIDT <i>width</i><br>[:SOUR]:DIG:EXT[n]:TOUT[:EDGE]:WIDT? [DEFault MINimum MAXimum]     | Sets the pulse width of the output trigger for the specified GPIO pin.<br><br><i>width</i> =MINimum MAXimum DEFault 1E-5 to 1E-2 seconds  | 1.00E-04 seconds |
| [:SOUR]:DIG:EXT[n]:TOUT:TYPE <i>type</i><br>[:SOUR]:DIG:EXT[n]:TOUT:TYPE?  | Selects the output trigger type for the specified GPIO pin.<br><br><i>type</i> =EDGE LEVel  | EDGE             |
| [:SOUR]:DIG:INT[i]:TOUT[:EDGE]:POS <i>position</i><br>[:SOUR]:DIG:INT[i]:TOUT[:EDGE]:POS?                              | Selects the trigger output timing for the internal trigger line 1 or 2.<br><br><i>position</i> =BEFore AFTer BOTH   | BOTH             |
| [:SOUR[c]]:FUNC:MODE <i>mode</i><br>[:SOUR[c]]:FUNC:MODE?  | Selects the source output mode of the specified channel.<br><br><i>mode</i> =CURRENT VOLTage  | VOLT             |
| [:SOUR[c]]:FUNC[:SHAP] <i>shape</i><br>[:SOUR[c]]:FUNC[:SHAP]?   | Selects the source output shape of the specified channel.<br><br><i>shape</i> =PULSe DC   | DC               |
| [:SOUR[c]]:FUNC:TRIG:CONT <i>mode</i><br>[:SOUR[c]]:FUNC:TRIG:CONT?  | Enables or disables continuous trigger output for the specified channel.<br><br><i>mode</i> =0 OFF 1 ON   | OFF              |
| [:SOUR[c]]:LIST:CURR <i>list</i><br>[:SOUR[c]]:LIST:VOLT <i>list</i><br>[:SOUR[c]]:LIST:CURR?<br>[:SOUR[c]]:LIST:VOLT? | Sets the list sweep output (current or voltage) data for the specified channel.<br><br><i>list</i> : List of the output data. <i>list</i> can have data up to maximum number of sweep steps (see “Sweep Steps” on page 2-28). Each data must be separated by a comma. | 0                |

| Command  | Summary  | Reset setting |
|--|--|---------------|
| <pre>[ :SOUR [c] ] :LIST:CURR:APP <i>append_list</i> [ :SOUR [c] ] :LIST:VOLT:APP <i>append_list</i></pre>   | <p>Adds the list sweep output (current or voltage) data to the end of the list set by the [:SOUR[c]]:LIST:&lt;CURR VOLT&gt; command, to which some data might be appended to by this command. The list can have data up to maximum number of sweep steps (see “Sweep Steps” on page 2-28).</p> <p><i>append_list</i>: List of the output data. Multiple data can be set to <i>append_list</i>. Each data must be separated by a comma.</p> |               |
| <pre>[ :SOUR [c] ] :LIST:CURR:POIN? [ :SOUR [c] ] :LIST:VOLT:POIN?</pre>   | <p>Returns the number of data in the list set by the [:SOUR[c]]:LIST:&lt;CURR VOLT&gt; command, to which some data might be appended to by the [:SOUR[c]]:LIST:&lt;CURR VOLT&gt;:APP command.</p>  |               |
| <pre>[ :SOUR [c] ] :LIST:CURR:STAR <i>start</i> [ :SOUR [c] ] :LIST:VOLT:STAR <i>start</i> [ :SOUR [c] ] :LIST:CURR:STAR? [ :SOUR [c] ] :LIST:VOLT:STAR?</pre> | <p>Specifies the list sweep start point by using the index of the list.</p> <p><i>start</i>: Index of the list. 1 to maximum number of sweep steps (see “Sweep Steps” on page 2-28). Other index value causes an error. <i>start</i>=1 indicates the first data in the list (top of the list).</p>   | 1             |
| <pre>[ :SOUR [c] ] :PULS:DEL <i>delay</i> [ :SOUR [c] ] :PULS:DEL? [DEFault MINimum MAXimum]</pre>   | <p>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.</p> <p><i>delay</i>=DEFault MINimum MAXimum  0.0 to 99999.9 seconds</p>   | 0             |

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary  | Reset setting |
|--|--|---------------|
| <code>[[:SOUR[c]]]:PULS:WIDT <i>width</i></code><br><code>[[:SOUR[c]]]:PULS:WIDT? [DEFault MINimum MAXimum]</code> | <p>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.</p> <p><i>width</i>=DEFault MINimum MAXimum <br/>5E-5 to 100000 seconds, in 1E-6 resolution</p> <p>Minimum time for the pulse base output is also 50 <math>\mu</math>s. And the minimum pulse period is 100 <math>\mu</math>s.</p>   | 5E-5 seconds  |
| <code>[[:SOUR[c]]]:SWE:DIR <i>direction</i></code><br><code>[[:SOUR[c]]]:SWE:DIR?</code>                           | <p>Sets the sweep direction, UP or DOWN, for the specified channel.</p> <p><i>direction</i>=DOWN UP</p>  | UP            |
| <code>[[:SOUR[c]]]:SWE:POIN <i>points</i></code><br><code>[[:SOUR[c]]]:SWE:POIN? DEFault MINimum MAXimum</code>    | <p>Sets the number of sweep steps for the specified channel. This command setting is effective for both current sweep and voltage sweep.</p> <p><i>points</i>=DEFault MINimum MAXimum or integer (see “Sweep Steps” on page 2-28)</p> <p>The points value can be expressed by the following formula, using the step value set by the<br/> <code>[[:SOUR[c]]]:&lt;CURR VOLT&gt;:STEP</code><br/> command and the span value set by the<br/> <code>[[:SOUR[c]]]:&lt;CURR VOLT&gt;:SPAN</code><br/> command.</p> <p><math>points = span/step + 1</math> (where <i>step</i> is not 0)</p> <p><i>points</i>=1 sets <i>step</i>=0.</p> | 1             |

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

Subsystem Command Summary  
Setting Source/Measure Unit

| Command   | Summary  | Reset setting |
|---|--|---------------|
| [:SOUR[c]]:WAIT:GAIN <i>gain</i><br>[:SOUR[c]]:WAIT:GAIN? [DEFault MINimum MAXimum]   | Sets the gain value used for calculating the source wait time for the specified channel.<br><br><i>gain</i> =MINimum MAXimum DEFault 0 to 100  | 1             |
| [:SOUR[c]]:WAIT:OFFS <i>offset</i><br>[:SOUR[c]]:WAIT:OFFS? [DEFault MINimum MAXimum] | Sets the offset value used for calculating the source wait time for the specified channel.<br><br><i>offset</i> =MINimum MAXimum DEFault 0 to 1 seconds  | 0             |
| [:SOUR[c]]:WAIT[:STAT] <i>mode</i><br>[: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.<br><br><i>mode</i> =0 or OFF disables the source wait time. The wait time is set to 0.<br><br><i>mode</i> =1 or ON enables the source wait time given by the following formula.<br><br><ul style="list-style-type: none"> <li>[:SOUR[c]]:WAIT:AUTO ON 1 condition:<br/><br/>wait time = <math>gain \times \text{initial wait time} + offset</math></li> <li>[:SOUR[c]]:WAIT:AUTO OFF 0 condition:<br/><br/>wait time = <i>offset</i></li> </ul> <p>The initial wait time is automatically set by the instrument and cannot be changed.</p> | ON            |



**Table 2-2 SENSE Subsystem**

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

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary  | Reset setting   |
|--|--|---|
| :SENS[c]:CURR[:DC]:NPLC:AUTO <i>mode</i><br>:SENS[c]:RES:NPLC:AUTO <i>mode</i><br>:SENS[c]:VOLT[:DC]:NPLC:AUTO <i>mode</i><br>:SENS[c]:CURR[:DC]:NPLC:AUTO?<br>:SENS[c]:RES:NPLC:AUTO?<br>:SENS[c]:VOLT[:DC]:NPLC:AUTO?        | <p>Enables or disables the automatic NPLC function.</p> <p><i>mode</i>=1 ON 0 OFF</p> <p>The automatic NPLC on/off works with the automatic aperture on/off set by the :SENS:&lt;CURR RES VOLT&gt;:APER:AUTO command. So the last command setting is effective for both functions.</p>   | OFF   |
| :SENS[c]:CURR[:DC]:PROT[:LEV][:BOTH] <i>comp</i><br>:SENS[c]:VOLT[:DC]:PROT[:LEV] <i>comp</i><br>:SENS[c]:CURR[:DC]:PROT[:LEV]? [DEFault MINimum MAXimum]<br>:SENS[c]:VOLT[:DC]:PROT[:LEV]? [DEFault MINimum MAXimum]          | <p>Sets the compliance value of the specified channel. The setting value is applied to both positive and negative sides.</p> <p>Keyword BOTH is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.</p> <p><i>comp</i>=MINimum MAXimum DEFault minimum to maximum measurement value, in A or V. See maximum current or maximum voltage in “<b>Source Output Ranges</b>” on page 2-24.</p> | 1.00E-04 A for current compliance, 2.0 V for voltage compliance   |
| :SENS[c]:CURR[:DC]:PROT[:LEV]:NEG <i>comp</i><br>:SENS[c]:VOLT[:DC]:PROT[:LEV]:NEG <i>comp</i><br>:SENS[c]:CURR[:DC]:PROT[:LEV]:NEG? [DEFault MINimum MAXimum]<br>:SENS[c]:VOLT[:DC]:PROT[:LEV]:NEG? [DEFault MINimum MAXimum] | <p>This command is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.</p> <p>Sets the negative side compliance value of the specified channel.</p> <p><i>comp</i>=MINimum MAXimum DEFault minimum to maximum measurement value, in A or V. See maximum current or maximum voltage in “<b>Source Output Ranges</b>” on page 2-24.</p>   | -1.00E-04 A for current compliance, -2.0 V for voltage compliance |

| Command   | Summary  | Reset setting   |
|---|--|---|
| :SENS[c]:CURR[:DC]:PROT[:LEV]:POS comp<br>:SENS[c]:VOLT[:DC]:PROT[:LEV]:POS comp<br>:SENS[c]:CURR[:DC]:PROT[:LEV]:POS? [DEFAULT MINimum MAXimum]<br>:SENS[c]:VOLT[:DC]:PROT[:LEV]:POS? [DEFAULT MINimum MAXimum]  | <p>This command is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.</p> <p>Sets the positive side compliance value of the specified channel.</p> <p>comp=MINimum MAXimum DEFAULT minimum to maximum measurement value, in A or V. See maximum current or maximum voltage in “Source Output Ranges” on page 2-24.</p> | +1.00E-04 A for current compliance, +2.0 V for voltage compliance |
| :SENS[c]:CURR[:DC]:PROT:TRIP?<br>:SENS[c]:VOLT[:DC]:PROT:TRIP?  | <p>Returns the compliance status of the specified channel</p> <p>Response is 1 or 0 that indicates the channel is in the compliance state or not.</p>  |   |
| :SENS[c]:CURR[:DC]:RANG:AUTO mode<br>:SENS[c]:RES:RANG:AUTO mode<br>:SENS[c]:VOLT[:DC]:RANG:AUTO mode<br>:SENS[c]:CURR[:DC]:RANG:AUTO?<br>:SENS[c]:RES:RANG:AUTO?<br>:SENS[c]:VOLT[:DC]:RANG:AUTO?  | <p>Enables or disables the automatic ranging function of the specified measurement channel.</p> <p>mode=1 ON 0 OFF</p>   | ON  |
| :SENS[c]:CURR[:DC]:RANG:AUTO:LLIM range<br>:SENS[c]:RES:RANG:AUTO:LLIM range<br>:SENS[c]:VOLT[:DC]:RANG:AUTO:LLIM range<br>:SENS[c]:CURR[:DC]:RANG:AUTO:LLIM? [DEFAULT MINimum MAXimum]<br>:SENS[c]:RES:RANG:AUTO:LLIM? [DEFAULT MINimum MAXimum]<br>:SENS[c]:VOLT[:DC]:RANG:AUTO:LLIM? [DEFAULT MINimum MAXimum] | <p>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.</p> <p>range=MINimum MAXimum DEFAULT minimum to maximum measurement value, in A, <math>\Omega</math>, or V. See “Measurement Ranges” on page 2-27.</p>   | 1.00E-06 A, 2 $\Omega$ , or 0.2 V                                 |

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary  | Reset setting  |
|--|--|----------------|
| :SENS[c]:CURR[:DC]:RANG:AUTO:MODE <i>mode</i><br>:SENS[c]:VOLT[:DC]:RANG:AUTO:MODE <i>mode</i><br>:SENS[c]:CURR[:DC]:RANG:AUTO:MODE?<br>:SENS[c]:VOLT[:DC]:RANG:AUTO:MODE?   | Selects the operation mode of the automatic ranging measurement. This command setting is not effective if the automatic ranging is disabled by the :SENS:<CURR VOLT>:RANG:AUTO command.<br><br><i>mode</i> =NORMal RESolution SPEed  | NORM           |
| :SENS[c]:CURR[:DC]:RANG:AUTO:THR <i>rate</i><br>:SENS[c]:VOLT[:DC]:RANG:AUTO:THR <i>rate</i><br>:SENS[c]:CURR[:DC]:RANG:AUTO:THR? [DEFault MINimum MAXimum]<br>:SENS[c]:VOLT[:DC]:RANG:AUTO:THR? [DEFault MINimum MAXimum] | Sets the threshold rate for the automatic measurement ranging operation.<br><br><i>rate</i> =MINimum MAXimum DEFault 11 % to 100 %   | 90             |
| :SENS[c]:RES:RANG:AUTO:ULIM <i>range</i><br>:SENS[c]:RES:RANG:AUTO:ULIM? [DEFault MINimum MAXimum]<br>:SENS[c]:CURR[:DC]:RANG:AUTO:ULIM?<br>: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.<br><br><i>range</i> =MINimum MAXimum DEFault value (see <a href="#">Table 2-9</a> ) | 200 M $\Omega$ |

| Command   | Summary   | Reset setting                      |
|---|---|------------------------------------|
| :SENS [c] :CURR [:DC] :RANG [:UPP] <i>range</i><br>:SENS [c] :RES :RANG [:UPP] <i>range</i><br>:SENS [c] :VOLT [:DC] :RANG [:UPP] <i>range</i><br>:SENS [c] :CURR [:DC] :RANG [:UPP] ? [DEFault MINimum MAXimum]<br>:SENS [c] :RES :RANG [:UPP] ? [DEFault MINimum MAXimum]<br>: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.<br><br><i>range</i> =UP DOWN MINimum MAXimum DEFault  minimum to maximum measurement value, in A, $\Omega$ , or V. See “Measurement Ranges” on page 2-27. | 1.00E-04 A, 20 k $\Omega$ , or 2 V |
| :SENS [c] :DATA? [ <i>offset</i> [, <i>size</i> ]]  | Returns the array data which contains all data for the element specified by the :FORM:ELEM:SENS command.<br><br><i>offset</i> =CURRENT START 0 to maximum<br><i>size</i> =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 <i>fctn</i> [, <i>fctn</i> [, <i>fctn</i> ]]<br>:SENS [c] :FUNC:OFF?  | Disables the specified measurement functions.<br><br><i>fctn</i> =“CURRENT[:DC]” “VOLTage[:DC]” “RESistance”  | “RES”                              |
| :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] <i>fctn</i> [, <i>fctn</i> [, <i>fctn</i> ]]<br>:SENS [c] :FUNC [:ON] ?   | Enables the specified measurement functions.<br><br><i>fctn</i> =“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.   |                                    |

Subsystem Command Summary  
Setting Source/Measure Unit

| Command  | Summary  | Reset setting |
|--|--|---------------|
| :SENS [c] :FUNC:STAT? <i>fctn</i>  | Returns if the specified measurement function is enabled or disabled.<br><br><i>fctn</i> ="CURRent[:DC]" "VOLTage[:DC]" "RESistance"   |               |
| :SENS [c] :REM <i>mode</i><br>:SENS [c] :REM?  | Enables or disables the remote sensing. Remote sensing must be enabled to use the 4-wire connection (Kelvin connection).<br><br><i>mode</i> =1 ON 0 OFF  | OFF           |
| :SENS [c] :RES:MODE <i>mode</i><br>:SENS [c] :RES:MODE?  | Selects the resistance measurement mode.<br><br><i>mode</i> =MANual  AUTO  | MAN           |
| :SENS [c] :RES:OCOM <i>mode</i><br>:SENS [c] :RES:OCOM?  | Enables or disables the offset-compensated resistance measurement.<br><br><i>mode</i> =1 ON 0 OFF  | OFF           |
| :SENS [c] :TOUT:SIGN <i>output{, output}</i><br>:SENS [c] :TOUT:SIGN <i>output{, output}</i><br>:SENS [c] :TOUT:SIGN <i>output{, output}</i><br>:SENS [c] :TOUT:SIGN?<br>:SENS [c] :TOUT:SIGN? | Selects the trigger output for the status change between the trigger layer and the acquire device action.<br><br><i>output</i> =INT1 INT2 LAN EXT1 EXT2 EXT3 EXT4 EXT5 EXT6 EXT7 EXT8 EXT9 EXT10 EXT11 EXT12 EXT13 EXT14 | EXT1          |
| :SENS [c] :TOUT[:STAT] <i>mode</i><br>:SENS [c] :TOUT[:STAT] <i>mode</i><br>:SENS [c] :TOUT[:STAT] <i>mode</i><br>:SENS [c] :TOUT[:STAT] ?<br>:SENS [c] :TOUT[:STAT] ?                         | Enables or disables the trigger output for the status change between the trigger layer and the acquire device action.<br><br><i>mode</i> =1 ON 0 OFF   | OFF           |
| :SENS [c] :WAIT:AUTO <i>mode</i><br>: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].<br><br><i>mode</i> =1 ON 0 OFF   | ON            |

| Command   | Summary   | Reset setting |
|---|---|---------------|
| :SENS[c]:WAIT:GAIN <i>gain</i><br>:SENS[c]:WAIT:GAIN? [DEFault MINimum MAXimum]   | Sets the gain value used for calculating the measurement wait time for the specified channel.<br><br><i>gain</i> =MINimum MAXimum DEFault 0 to 100  | 1             |
| :SENS[c]:WAIT:OFFS <i>offset</i><br>:SENS[c]:WAIT:OFFS? [DEFault MINimum MAXimum] | Sets the offset value used for calculating the measurement wait time for the specified channel.<br><br><i>gain</i> =MINimum MAXimum DEFault 0 to 1 seconds  | 0             |
| :SENS[c]:WAIT[:STAT] <i>mode</i><br>: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.<br><br><i>mode</i> =0 or OFF disables the measurement wait time. The wait time is set to 0.<br><br><i>mode</i> =1 or ON enables the measurement wait time given by the following formula.<br><br><ul style="list-style-type: none"> <li>:SENS[c]:WAIT:AUTO ON 1 condition:<br/><math display="block">\text{wait time} = \text{gain} \times \text{initial wait time} + \text{offset}</math></li> <li>:SENS[c]:WAIT:AUTO OFF 0 condition:<br/><math display="block">\text{wait time} = \text{offset}</math></li> </ul> The initial wait time is automatically set by the instrument and cannot be changed. | ON            |

Subsystem Command Summary  
Setting Source/Measure Unit

**Table 2-3**                      **OUTPut Subsystem**

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :OUTP[c]:FILT:AUTO <i>mode</i><br>:OUTP[c]:FILT:AUTO?   | Enables or disables the automatic filter function.<br><br><i>mode</i> =1 ON 0 OFF  | OFF           |
| :OUTP[c]:FILT[:LPAS]:FREQ <i>freq</i><br>:OUTP[c]:FILT[:LPAS]:FREQ? [DEFa<br>ult 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.<br><br><i>freq</i> =MINimum MAXimum DEFAULT <br>31.830 Hz to +31.831 kHz<br><br>$freq = 1/(2 \times \pi \times Tconst)$                                    | MIN           |
| :OUTP[c]:FILT[:LPAS][:STAT] <i>mode</i><br>:OUTP[c]:FILT[:LPAS][:STAT]?                             | Enables or disables the output filter.<br><br><i>mode</i> =1 ON 0 OFF  | ON            |
| :OUTP[c]:FILT[:LPAS]:TCON <i>Tconst</i><br>:OUTP[c]:FILT[:LPAS]:TCON? [DEFa<br>ult 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.<br><br><i>Tconst</i> =MINimum MAXimum DEFAULT <br>5 $\mu$ s to 5 ms<br><br>$Tconst = 1/(2 \times \pi \times freq)$ | MIN           |
| :OUTP[c]:HCAP[:STAT] <i>mode</i><br>:OUTP[c]:HCAP[:STAT]?   | Enables or disables the high capacitance mode.<br><br><i>mode</i> =1 ON 0 OFF  | OFF           |
| :OUTP[c]:LOW <i>low_state</i><br>:OUTP[c]:LOW?  | Selects the state of the low terminal.<br><br><i>low_state</i> =FLOat GROund   | GRO           |
| :OUTP[c]:OFF:AUTO <i>mode</i><br>:OUTP[c]:OFF:AUTO?   | Enables or disables the automatic output off function.<br><br><i>mode</i> =1 ON 0 OFF  | OFF           |



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

## Source Output Ranges

**Table 2-4** Voltage Output Range

| Range<br>value | Setting resolution |                  | DC output voltage<br>or pulse peak/base<br>voltage | Maximum current <sup>a</sup> |  | Pulse width t <sup>b</sup>   |  |
|----------------|--------------------|------------------|--|------------------------------|--|------------------------------|--|
|                | B2901A<br>B2902A   | B2911A<br>B2912A |  | DC<br>output                 | Pulsed<br>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 <sub>max</sub>                                  |                              |  |
| 2 V            | 10 μV              | 1 μV             | 0 ≤  V  ≤ 2.1 V                                    |                              | ±10.5 A with 50 μs ≤ t ≤ 1 ms  |                              |  |
| 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 <sub>max</sub>                                 |                              |  |
| 200 V          | 1 mV               | 100 μV           | 0 ≤  V  ≤ 6 V                                      | ±3.03 A                      | ±3.03 A with 50 μs ≤ t ≤ t <sub>max</sub><br>±10.5 A with 50 μs ≤ t ≤ 1 ms |                              |  |
|                |                    |                  | 6 V <  V  ≤ 21 V                                   | ±1.515 A                     | ±1.515 A   | 50 μs ≤ t ≤ t <sub>max</sub> |  |
|                |                    |                  | 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  $50 \mu\text{s} \leq t \leq t_{\text{max}}$  ( $=99.9999$  ks).
- b. Maximum duty cycle is 99.9999 % for the pulse with  $50 \mu\text{s} \leq t \leq t_{\text{max}}$ , and 2.5 % for the pulse with  $50 \mu\text{s} \leq t \leq 1$  ms,  $50 \mu\text{s} \leq t \leq 2.5$  ms, or  $50 \mu\text{s} \leq t \leq 10$  ms.

**Table 2-5** Limitations for using Channels 1 and 2

| Channel 1 voltage V1           | Channel 2 voltage V2           | Current limit <sup>a</sup>        |
|--------------------------------|--------------------------------|-----------------------------------|
| $0 <  V1  \leq 6$ V            | $0 <  V2  \leq 6$ V            | $I1 + I2 \leq 4$ A                |
|                                | $6 \text{ V} <  V2  \leq 21$ V | $I1 + I2 \times 1.6 \leq 4$ A     |
| $6 \text{ V} <  V1  \leq 21$ V | $0 <  V2  \leq 6$ V            | $I1 + I2 \times 0.625 \leq 2.5$ A |
|                                | $6 \text{ V} <  V2  \leq 21$ V | $I1 + I2 \leq 2.5$ A              |

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

**Table 2-6 Current Output Range**

| Range<br>value     | Setting resolution |                  | DC output current or<br>pulse peak/base<br>current <sup>a b</sup> | Maximum voltage |                  | Pulse width t <sup>c</sup>   |
|--------------------|--------------------|------------------|---|-----------------|------------------|------------------------------|
|                    | B2901A<br>B2902A   | B2911A<br>B2912A |   | DC<br>output    | Pulsed<br>output |                              |
| 10 nA <sup>d</sup> | —                  | 10 fA            | 0 ≤  I  ≤ 10.5 nA   | ±210 V          | ±210 V           | 50 μs ≤ t ≤ t <sub>max</sub> |
| 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  | ±21 V           | ±21 V            |                              |
|                    |                    |                  | 105 mA <  I  ≤ 1.05 A   |                 |                  |                              |
|                    |                    |                  | 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 <sub>max</sub> |
|                    |                    |                  | 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 <sub>max</sub> |
|                    |                    |                  | 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            |

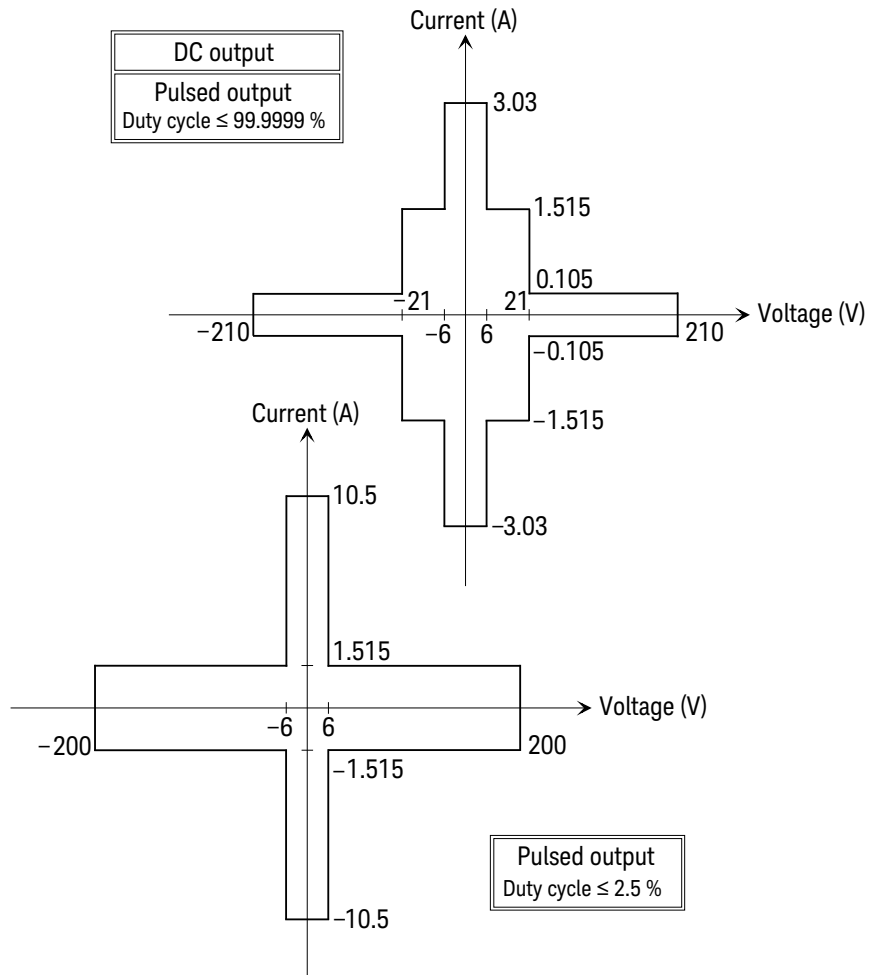
# Subsystem Command Summary

## Setting Source/Measure Unit

- Table 2-5** shows the limitations when using Channels 1 and 2 for DC output or Pulsed output with  $50 \mu\text{s} \leq t \leq t_{\text{max}}$  ( $=99.9999 \text{ ks}$ ).
- Maximum base current is 500 mA for the pulse with  $50 \mu\text{s} \leq t \leq 1 \text{ ms}$ , and 50 ms for the pulse with  $50 \mu\text{s} \leq t \leq 2.5 \text{ ms}$  or  $50 \mu\text{s} \leq t \leq 10 \text{ ms}$ .
- Maximum duty cycle is 99.9999 % for the pulse with  $50 \mu\text{s} \leq t \leq t_{\text{max}}$ , and 2.5 % for the pulse with  $50 \mu\text{s} \leq t \leq 1 \text{ ms}$ ,  $50 \mu\text{s} \leq t \leq 2.5 \text{ ms}$ , or  $50 \mu\text{s} \leq t \leq 10 \text{ ms}$ .
- 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       | $0 \leq  V  \leq 0.212 \text{ V}$ | 0.1 $\mu\text{V}$ |
| 2 V         | $0 \leq  V  \leq 2.12 \text{ V}$  | 1 $\mu\text{V}$   |
| 20 V        | $0 \leq  V  \leq 21.2 \text{ V}$  | 10 $\mu\text{V}$  |
| 200 V       | $0 \leq  V  \leq 212 \text{ V}$   | 100 $\mu\text{V}$ |

Table 2-8

Current Measurement Range

| Range value        | Current measurement value          | Resolution       |
|--------------------|------------------------------------|------------------|
| 10 nA <sup>a</sup> | $0 \leq  I  \leq 10.6 \text{ nA}$  | 10 fA            |
| 100 nA             | $0 \leq  I  \leq 106 \text{ nA}$   | 100 fA           |
| 1 $\mu\text{A}$    | $0 \leq  I  \leq 1.06 \mu\text{A}$ | 1 pA             |
| 10 $\mu\text{A}$   | $0 \leq  I  \leq 10.6 \mu\text{A}$ | 10 pA            |
| 100 $\mu\text{A}$  | $0 \leq  I  \leq 106 \mu\text{A}$  | 100 pA           |
| 1 mA               | $0 \leq  I  \leq 1.06 \text{ mA}$  | 1 nA             |
| 10 mA              | $0 \leq  I  \leq 10.6 \text{ mA}$  | 10 nA            |
| 100 mA             | $0 \leq  I  \leq 106 \text{ mA}$   | 100 nA           |
| 1 A                | $0 \leq  I  \leq 1.06 \text{ A}$   | 1 $\mu\text{A}$  |
| 1.5 A              | $0 \leq  I  \leq 1.53 \text{ A}$   |                  |
| 3 A                | $0 \leq  I  \leq 3.06 \text{ A}$   | 10 $\mu\text{A}$ |
| 10 A <sup>b</sup>  | $0 \leq  I  \leq 10.6 \text{ A}$   |                  |

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<sup>1</sup>**

| Range value    | Resistance measurement value                      | Display resolution | Test current      |
|----------------|---|--------------------|-------------------|
| 2 $\Omega$     | $0 < R \leq 2 \Omega$                             | 1 $\mu\Omega$      | 1A                |
| 20 $\Omega$    | $2 \Omega < R \leq 20 \Omega$                     | 10 $\mu\Omega$     | 100 mA            |
| 200 $\Omega$   | $20 \Omega < R \leq 200 \Omega$                   | 100 $\mu\Omega$    | 10 mA             |
| 2 k $\Omega$   | $200 \Omega < R \leq 2 \text{ k}\Omega$           | 1 m $\Omega$       | 1 mA              |
| 20 k $\Omega$  | $2 \text{ k}\Omega < R \leq 20 \text{ k}\Omega$   | 10 m $\Omega$      | 100 $\mu\text{A}$ |
| 200 k $\Omega$ | $20 \text{ k}\Omega < R \leq 200 \text{ k}\Omega$ | 100 m $\Omega$     | 10 $\mu\text{A}$  |
| 2 M $\Omega$   | $200 \text{ k}\Omega < R \leq 2 \text{ M}\Omega$  | 1 $\Omega$         | 1 $\mu\text{A}$   |
| 20 M $\Omega$  | $2 \text{ M}\Omega < R \leq 20 \text{ M}\Omega$   | 10 $\Omega$        | 100 nA            |
| 200 M $\Omega$ | $20 \text{ M}\Omega < R \leq 200 \text{ M}\Omega$ | 100 $\Omega$       | 10 nA             |

## Sweep Steps

Available number of sweep steps is 1 to 100000 for the latest B2900. However it is 1 to 2500 for the following B2900.

- B2900 series without the license “SWS”
- B2900 series with the firmware revision 3.0 or earlier
- B2901A with the serial number MY51140470 or below
- B2902A with the serial number MY51140683 or below
- B2911A with the serial number MY51140213 or below
- B2912A with the serial number MY51140498 or below

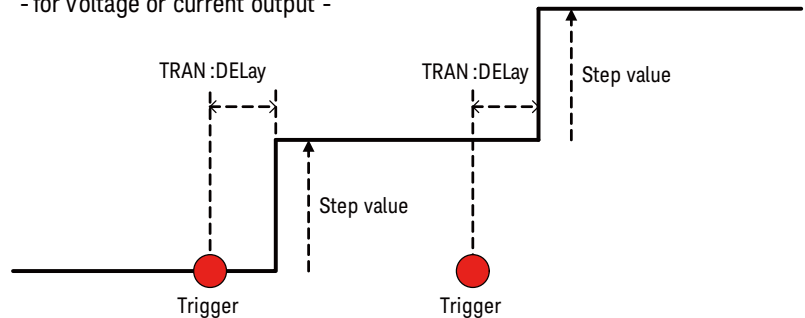
1. The resistance measurement range is effective for the resistance measurements set to the AUTO mode which is selected by the **:SENSe:RESistance:MODE** command.

## Controlling Source/Measure Trigger

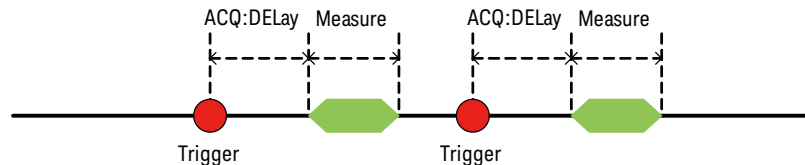
Figure 2-6

### Transient and Acquire Device Actions

TRANSient device action  
- for voltage or current output -



ACQure device action  
- for voltage or current measurement -



### NOTE

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

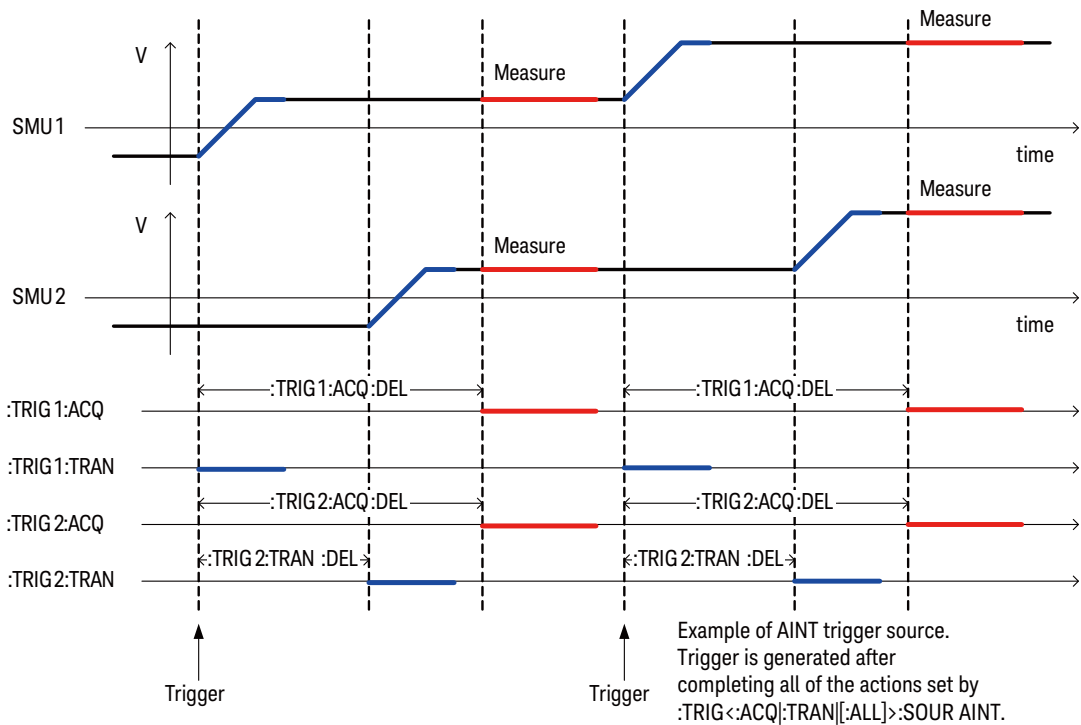
- To synchronize transient actions (source output)
  - Trigger source is set to the same mode.
  - Delay time is set to the same value.
  - Source output ranging mode is set to the fixed mode.
  - Source wait time control is set to OFF.
  - Measurement wait time control is set to OFF.
  - Measurement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

## Subsystem Command Summary

### Controlling Source/Measure Trigger

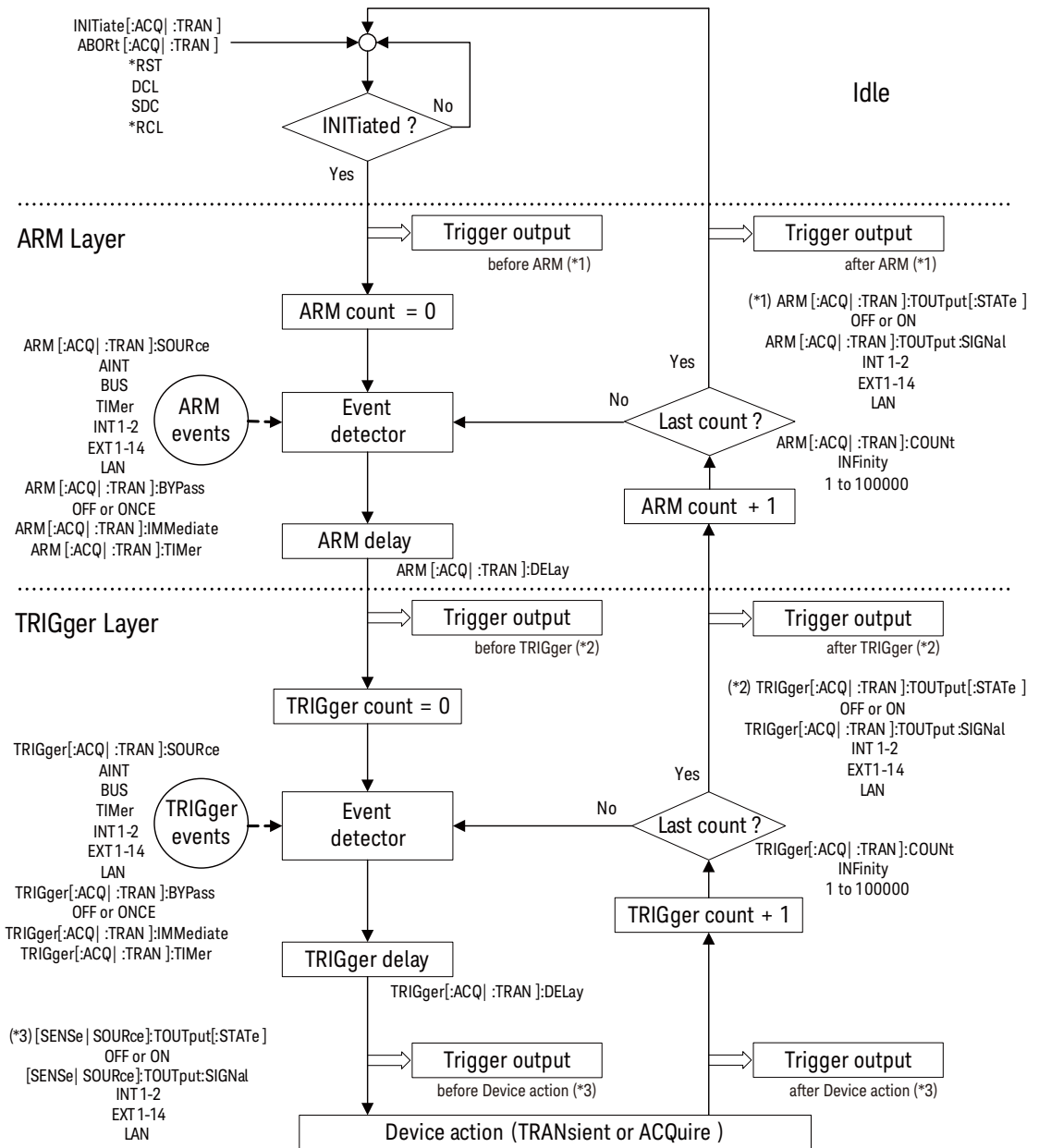
- To synchronize acquire actions (measurement)
  - Trigger source is set to the same mode.
  - Delay time is set to the same value.
  - Measurement wait time control is set to OFF.
  - Measurement ranging mode is set to the fixed mode.

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





**Figure 2-8 B2900 Trigger System**



Subsystem Command Summary  
Controlling Source/Measure Trigger

**Table 2-10 TRIGger Subsystem**

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

| Command   | Summary  | Reset setting    |
|---|--|------------------|
| :ARM[c]:ACQ[:LAY]:DEL <i>delay</i><br>:ARM[c]:TRAN[:LAY]:DEL <i>delay</i><br>:ARM[c][:ALL][:LAY]:DEL <i>delay</i><br>:ARM[c]:ACQ[:LAY]:DEL? [ <i>delay</i> ]<br>:ARM[c]:TRAN[:LAY]:DEL? [ <i>delay</i> ]<br>:ARM[c][:ALL][:LAY]:DEL? <i>delay</i>                       | Sets the arm delay for the specified device action.<br><br><i>delay</i> =MINimum MAXimum DEFault 0 to 100000 seconds<br><br>Query does not support <i>delay</i> =0 to 100000.                                | 0                |
| :ARM[c]:ACQ[:LAY]:SOUR:LAN <i>lan_id</i> {<br>, <i>lan_id</i> }<br>:ARM[c]:TRAN[:LAY]:SOUR:LAN <i>lan_id</i> {<br>, <i>lan_id</i> }<br>:ARM[c][:ALL][:LAY]:SOUR:LAN <i>lan_id</i> {<br>, <i>lan_id</i> }<br>:ARM[c]:ACQ[:LAY]:SOUR:LAN?<br>:ARM[c]:TRAN[:LAY]:SOUR:LAN? | Specifies one or more LXI triggers used for the arm source for the specified device action.<br><br><i>lan_id</i> =LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7  | All is selected. |
| :ARM[c]:ACQ[:LAY]:SOUR[:SIGN] <i>source</i><br>:ARM[c]:TRAN[:LAY]:SOUR[:SIGN] <i>source</i><br>:ARM[c][:ALL][:LAY]:SOUR[:SIGN] <i>source</i><br>:ARM[c]:ACQ[:LAY]:SOUR[:SIGN]?<br>:ARM[c]:TRAN[:LAY]:SOUR[:SIGN]?   | Selects the arm source for the specified device action.<br><br><i>source</i> =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 <i>interval</i><br>:ARM[c]:TRAN[:LAY]:TIM <i>interval</i><br>:ARM[c][:ALL][:LAY]:TIM <i>interval</i><br>:ARM[c]:ACQ[:LAY]:TIM? [ <i>interval</i> ]<br>:ARM[c]:TRAN[:LAY]:TIM? [ <i>interval</i> ]<br>:ARM[c][:ALL][:LAY]:TIM? <i>interval</i>     | Sets the interval of the TIMer arm source for the specified device action.<br><br><i>interval</i> =MINimum MAXimum DEFault 1E-5 to 1E+5 seconds<br><br>Query does not support <i>interval</i> =1E-5 to 1E+5. | 1E-5 seconds     |

Subsystem Command Summary  
Controlling Source/Measure Trigger

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

| Command  | Summary   | Reset setting  |
|--|---|--|
| :TRIG[c]:ACQ:COUN <i>count</i><br>:TRIG[c]:TRAN:COUN <i>count</i><br>:TRIG[c][:ALL]:COUN <i>count</i><br>:TRIG[c]:ACQ:COUN? [ <i>count</i> ]<br>:TRIG[c]:TRAN:COUN? [ <i>count</i> ]<br>:TRIG[c][:ALL]:COUN? <i>count</i>  | Sets the trigger count for the specified device action.<br><br><i>count</i> =INFinity MINimum MAXimum DEFault  1 to 100000 or 2147483647<br><br><i>count</i> =2147483647 indicates infinity.<br><br>Query does not support <i>count</i> =INFinity, 1 to 100000 and 2147483647.<br><br><i>Arm count</i> × <i>Trigger count</i> must be less than 100001. | 1  |
| :TRIG[c]:ACQ:DEL <i>delay</i><br>:TRIG[c]:TRAN:DEL <i>delay</i><br>:TRIG[c][:ALL]:DEL <i>delay</i><br>:TRIG[c]:ACQ:DEL? [ <i>delay</i> ]<br>:TRIG[c]:TRAN:DEL? [ <i>delay</i> ]<br>:TRIG[c][:ALL]:DEL? <i>delay</i>  | Sets the trigger delay for the specified device action.<br><br><i>delay</i> =MINimum MAXimum DEFault  0 to 100000 seconds<br><br>Query does not support <i>delay</i> =0 to 100000.  | 0  |
| :TRIG:ACQ[:IMM] [ <i>chanlist</i> ]<br>:TRIG:TRAN[:IMM] [ <i>chanlist</i> ]<br>:TRIG[:ALL][:IMM] [ <i>chanlist</i> ]   | 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.<br><br><i>chanlist</i> =( <i>@1</i> ) ( <i>@2</i> ) ( <i>@1,2</i> ) ( <i>@1:2</i> ) ( <i>@2,1</i> ) ( <i>@2:1</i> )                                   | ( <i>@1</i> ) for 1-ch models<br>( <i>@1,2</i> ) for 2-ch models |
| :TRIG[c]:ACQ[:LAY]:SOUR:LAN <i>lan_id</i> {, <i>lan_id</i> }<br>:TRIG[c]:TRAN[:LAY]:SOUR:LAN <i>lan_id</i> {, <i>lan_id</i> }<br>:TRIG[c][:ALL][:LAY]:SOUR:LAN <i>lan_id</i> {, <i>lan_id</i> }<br>:TRIG[c]:ACQ[:LAY]:SOUR:LAN?<br>:TRIG[c]:TRAN[:LAY]:SOUR:LAN? | Specifies one or more LXI triggers used for the trigger source for the specified device action.<br><br><i>lan_id</i> =LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7   | All is selected.   |

Subsystem Command Summary  
Controlling Source/Measure Trigger

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

## LXI Trigger Events

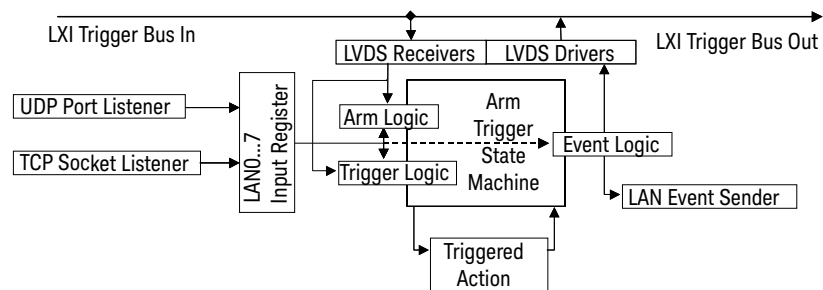
B2900 provides a subset of the LXI Trigger Events (IVI-3.15 IviLxiSync) functionality in the trigger system.

- Device Model

**Figure 2-9** 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 LANn ( $n$ : 0 to 7) trigger events by the instrument specific trigger event, slope, drive logic, destination, and filter.

**Figure 2-9 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

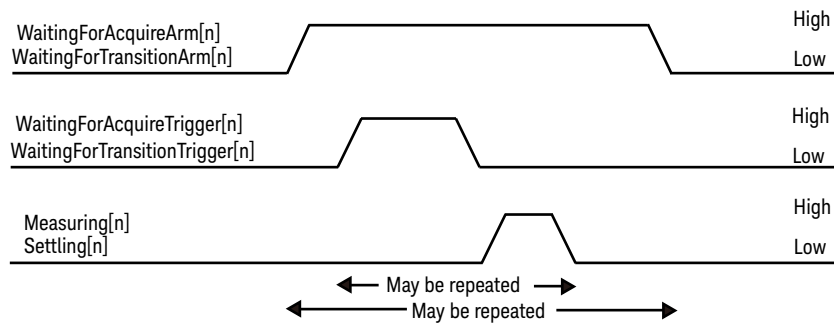
## Subsystem Command Summary

### Controlling Source/Measure Trigger

- Measuring2
- Settling1
- Settling2

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

**Figure 2-10 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)



**Table 2-11 LXI Subsystem**

| Command  | Summary   | Reset setting |
|--|---|---------------|
| :ARM:LXI:COUN <i>intRepetitions</i><br>:ARM:LXI:COUN?                                | This command is ignored.  | 0             |
| :ARM:LXI:DEL <i>delay</i><br>:ARM:LXI:DEL?   | Delay time must be zero.<br><i>delay</i> : Delay time, in seconds   | 0             |
| :ARM:LXI:LAN[:SET]:DET <i>event,detect</i><br>:ARM:LXI:LAN[:SET]:DET? <i>event</i>   | Specifies the style of arm source detection for the specified event.<br><i>event</i> : LAN event name<br><i>detect</i> =RISE FALL HIGH LOW  | RISE          |
| :ARM:LXI:LAN[:SET]:ENAB <i>event,status</i><br>:ARM:LXI:LAN[:SET]:ENAB? <i>event</i> | Enables or disables the arm source of the specified event.<br><i>event</i> : LAN event name<br><i>status</i> =0 OFF 1 ON  | OFF           |
| :ARM:LXI:LAN[:SET]:FILT <i>event,filter</i><br>:ARM:LXI:LAN[:SET]:FILT? <i>event</i> | Specifies a filter for restricting arm sources of the specified event.<br><i>event</i> : LAN event name<br><i>filter</i> : Filter. It will be an existing LAN arm sources, e.g. one of the items returned by :LXI:EVEN:INP:LAN:LIST? or :LXI:EVEN[:OUTP]:LAN:LIST?. | "ALL:5044"    |
| :ARM:LXI:LAN[:SET]:IDEN <i>event,id</i><br>:ARM:LXI:LAN[:SET]:IDEN? <i>event</i>     | Specifies the LAN event identifier that is associated with this arm source.<br><i>event</i> : LAN event name<br><i>id</i> : Identifier  | <i>event</i>  |
| :LXI:EVEN:DOM <i>domain</i><br>:LXI:EVEN:DOM?  | Specifies the LXI LAN domain.<br><i>domain</i> =0 to 255  | 0             |
| :LXI:EVEN:INP:LAN:ADD <i>event</i>   | This command is ignored.  |               |

Subsystem Command Summary  
Controlling Source/Measure Trigger

| Command  | Summary  | Reset setting |
|--|--|---------------|
| :LXI:EVEN:INP:LAN:COUN?  | Returns an integer as the total number of defined input LAN events (includes both enabled and disabled events).  | 8             |
| :LXI:EVEN:INP:LAN:DIS:ALL  | Disables all input events.   |               |
| :LXI:EVEN:INP:LAN:LIST?  | Returns a quoted string with the list of defined input event names.  |               |
| :LXI:EVEN:INP:LAN:REM:ALL  | This command is ignored.   |               |
| :LXI:EVEN:INP:LAN:REM <i>event</i>   | This command is ignored.   |               |
| :LXI:EVEN:INP:LAN[:SET]:CONF <i>event</i> ,<br><i>status</i> , <i>detect</i> [, <i>delay</i> ], <i>filter</i> ,<br><i>id</i> | Configures the most common attributes of LXI LAN input events.<br><br><i>event</i> : LAN event name<br><br><i>status</i> : Status<br><br><i>detect</i> : Detection type<br><br><i>delay</i> : Delay time<br><br><i>filter</i> : Filter<br><br><i>id</i> : Identifier |               |
| :LXI:EVEN:INP:LAN[:SET]:DEL <i>event</i> ,<br><i>delay</i><br>:LXI:EVEN:INP:LAN[:SET]:DEL? <i>event</i><br><i>nt</i>         | Delay time must be zero.<br><br><i>event</i> : LAN event name<br><br><i>delay</i> : Delay time, in seconds   | 0             |
| :LXI:EVEN:INP:LAN[:SET]:DET <i>event</i> ,<br><i>detect</i><br>:LXI:EVEN:INP:LAN[:SET]:DET? <i>event</i><br><i>nt</i>        | Specifies the trigger detection method and polarity for the input event.<br><br><i>event</i> : LAN event name<br><br><i>detect</i> =RISE FALL HIGH LOW   | RISE          |
| :LXI:EVEN:INP:LAN[:SET]:ENAB <i>event</i> ,<br><i>status</i><br>:LXI:EVEN:INP:LAN[:SET]:ENAB? <i>event</i><br><i>nt</i>      | Enables or disables the specified input event.<br><br><i>event</i> : LAN event name<br><br><i>status</i> =0 OFF 1 ON   | OFF           |

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :LXI:EVEN:INP:LAN[:SET]:FILT <i>event, filter</i><br>:LXI:EVEN:INP:LAN[:SET]:FILT? <i>event</i> | Creates a filter for incoming input events.<br><i>event</i> : LAN event name<br><i>filter</i> : Filter. It will be an existing LAN event, e.g. one of the items returned by :LXI:EVEN:INP:LAN:LIST?. | "ALL:5044"    |
| :LXI:EVEN:INP:LAN[:SET]:IDEN <i>event, id</i><br>:LXI:EVEN:INP:LAN[:SET]:IDEN? <i>event</i>     | Specifies the string that is expected to arrive over the LAN for a given input event to occur.<br><i>event</i> : LAN event name<br><i>id</i> : Identifier  | <i>event</i>  |
| :LXI:EVEN:LOG:ALL?  | Returns the contents of the event log.   |               |
| :LXI:EVEN:LOG:CIRC[:ENAB] <i>status</i><br>:LXI:EVEN:LOG:CIRC[:ENAB]?                           | Selects how new entries are handled when the LXI event log is full.<br><i>status</i> =0 OFF 1 ON   | ON            |
| :LXI:EVEN:LOG:CIRC:FBE  | Selects the most recently added event log entry to be used as the reference for :LXI:EVEN:LOG:ENTR?.   |               |
| :LXI:EVEN:LOG:CLE   | Removes all existing entries from the event log.   |               |
| :LXI:EVEN:LOG:COUN?   | Returns an integer as the total number of entries in the LXI event log.  |               |
| :LXI:EVEN:LOG:ENAB <i>status</i><br>:LXI:EVEN:LOG:ENAB?   | Enables or disables LXI event logging.<br><i>status</i> =0 OFF 1 ON  | ON            |
| :LXI:EVEN:LOG:ENTR? <i>index</i>  | Retrieves the event log entry referenced by <i>index</i> .<br><i>index</i> =0 to 2147483647  |               |
| :LXI:EVEN:LOG:SIZE <i>size</i><br>:LXI:EVEN:LOG:SIZE?   | Sets the maximum number of entries the LXI event log can hold.<br><i>size</i> =0 to 200  | 100           |
| :LXI:EVEN[:OUTP]:LAN:ADD <i>event</i>   | This command is ignored.   |               |

Subsystem Command Summary  
Controlling Source/Measure Trigger

| Command   | Summary   | Reset setting |
|---|---|---------------|
| :LXI:EVEN[:OUTP]:LAN:COUN?  | Returns an integer as the number of configured LXI output LAN events.   |               |
| :LXI:EVEN[:OUTP]:LAN:DIS:ALL  | Disables all configured LXI output LAN events.  |               |
| :LXI:EVEN[:OUTP]:LAN:LIST?  | Returns a quoted string containing a list of all configured LAN output event names.   |               |
| :LXI:EVEN[:OUTP]:LAN:REM:ALL  | This command is ignored.  |               |
| :LXI:EVEN[:OUTP]:LAN:REM <i>event</i>   | This command is ignored.  |               |
| :LXI:EVEN[:OUTP]:LAN:SEND <i>event</i> ,<br><i>type</i>   | Forces the instrument to send the specified output event.<br><br><i>event</i> : LAN event name<br><br><i>type</i> =RISE FALL  |               |
| :LXI:EVEN[:OUTP]:LAN[:SET]:CONF<br><i>event</i> , <i>status</i> , <i>source</i> , <i>slope</i> , <i>drive</i> ,<br><i>destination</i> | Configures the most common attributes of LXI LAN output events.<br><br><i>event</i> : LAN event name<br><br><i>status</i> : Status<br><br><i>source</i> : Event name<br><br><i>slope</i> : Slope attribute<br><br><i>drive</i> : Drive behavior<br><br><i>destination</i> : Destination |               |
| :LXI:EVEN[:OUTP]:LAN[:SET]:DEST<br><i>event</i> , <i>destination</i><br><br>:LXI:EVEN[:OUTP]:LAN[:SET]:DEST?<br><i>event</i>          | Sets the destination for the specified outgoing LAN event to the hosts specified by <i>destination</i> .<br><br><i>event</i> : LAN event name<br><br><i>destination</i> : Destination   | "ALL:5044"    |

| Command   | Summary  | Reset setting |
|---|--|---------------|
| : LXI : EVEN [ : OUTP ] : LAN [ : SET ] : DRIV<br>event, drive<br>: LXI : EVEN [ : OUTP ] : LAN [ : SET ] : DRIV?<br>event    | Specifies the trigger drive behavior for the specified LAN output event.<br>event: LAN event name<br>drive=OFF NORMal WOR  | OFF           |
| : LXI : EVEN [ : OUTP ] : LAN [ : SET ] : ENAB<br>event, status<br>: LXI : EVEN [ : OUTP ] : LAN [ : SET ] : ENAB?<br>event   | Enables or disables the specified LXI LAN output event.<br>event: LAN event name<br>status=0 OFF 1 ON  | OFF           |
| : LXI : EVEN [ : OUTP ] : LAN [ : SET ] : IDEN<br>event, id<br>: LXI : EVEN [ : OUTP ] : LAN [ : SET ] : IDEN?<br>event       | Specifies the custom string that will be transmitted as part of the output event.<br>event: LAN event name<br>id: Identifier   | event         |
| : LXI : EVEN [ : OUTP ] : LAN [ : SET ] : SLOP<br>event, slope<br>: LXI : EVEN [ : OUTP ] : LAN [ : SET ] : SLOP?<br>event    | Sets the slope of the event transition.<br>event: LAN event name<br>slope=POSitive NEGative  | POS           |
| : LXI : EVEN [ : OUTP ] : LAN [ : SET ] : SOUR<br>event, anyEvent<br>: LXI : EVEN [ : OUTP ] : LAN [ : SET ] : SOUR?<br>event | Designates the instrument that the specified LAN output event is tied to.<br>event: LAN event name<br>anyEvent: One of the following event names.<br>WaitingForAcquireArm1,<br>WaitingForAcquireArm2,<br>WaitingForAcquireTrigger1,<br>WaitingForAcquireTrigger2,<br>WaitingForTransitionArm1,<br>WaitingForTransitionArm2,<br>WaitingForTransitionTrigger1,<br>WaitingForTransitionTrigger2,<br>Measuring1, Measuring2, Settling1, or Settling2 | ""            |

Subsystem Command Summary  
Controlling Source/Measure Trigger

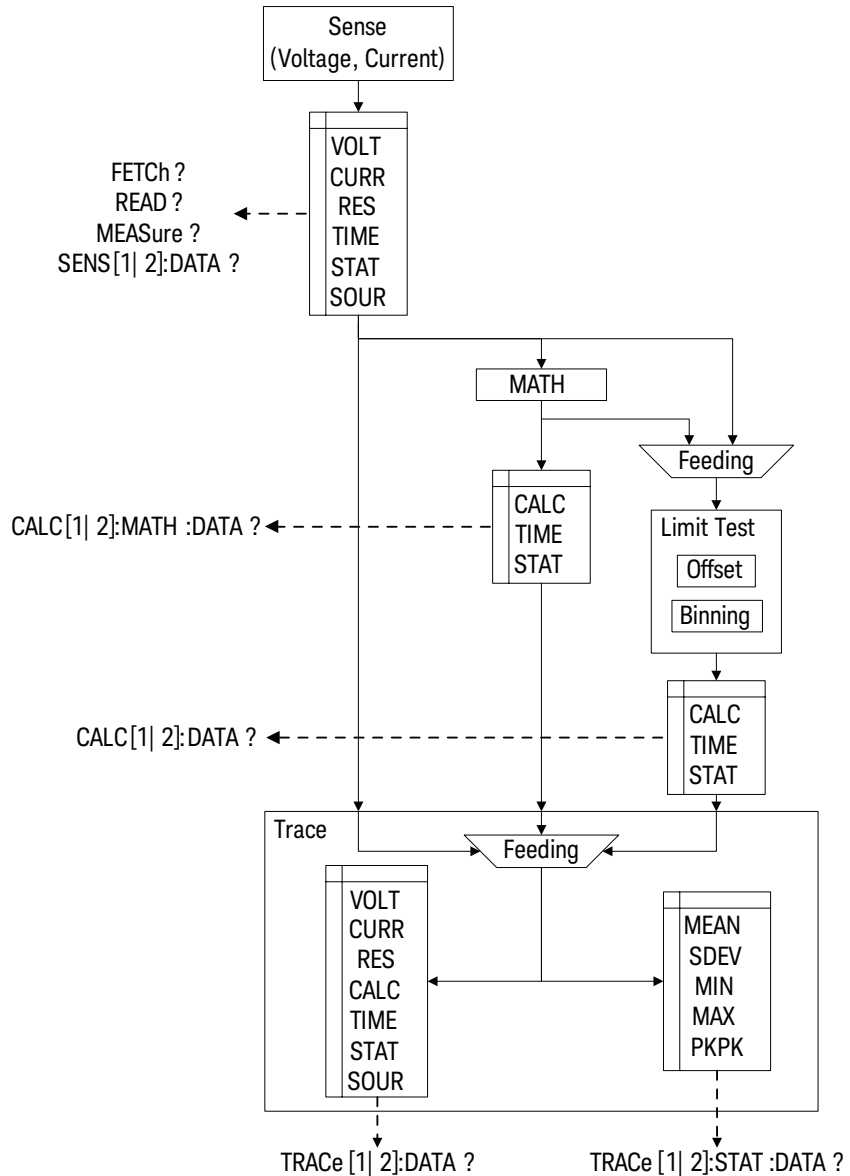
| Command  | Summary   | Reset setting   |
|--|---|-----------------|
| <code>:LXI:EVEN[:OUTP]:LAN[:SET]:TSD event, delay</code><br><code>:LXI:EVEN[:OUTP]:LAN[:SET]:TSD? event</code> | Sets the delay that occurs between the generation of the specified event and the remote instruments action on it.<br>The delay time must be zero.<br><i>event</i> : LAN event name<br><i>delay</i> : Delay time, in seconds | 0               |
| <code>:LXI:IDEN[:STAT] mode</code><br><code>:LXI:IDEN[:STAT]?</code>   | Changes the LXI status indicator state.<br><i>mode</i> =0 OFF 1 ON  | OFF             |
| <code>:LXI:MDNS:ENAB mode</code><br><code>:LXI:MDNS:ENAB?</code>   | Enables or disables mDNS (multicast DNS) function.<br><i>mode</i> =0 OFF 1 ON   | ON              |
| <code>:LXI:MDNS:HNAM[:RES]?</code>   | Returns the resolved mDNS hostname.   |                 |
| <code>:LXI:MDNS:SNAM:DES name</code><br><code>:LXI:MDNS:SNAM:DES?</code>                                       | Sets the desired mDNS service name.<br><i>name</i> : Desired mDNS service name  | "A-B29xxA-xxxx" |
| <code>:LXI:MDNS:SNAM[:RES]?</code>   | Returns the resolved mDNS service name.   |                 |
| <code>:TRIG:LXI:LAN[:SET]:DEL delay</code><br><code>:TRIG:LXI:LAN[:SET]:DEL?</code>                            | Delay time must be zero.<br><i>delay</i> : Delay time, in seconds   | 0               |
| <code>:TRIG:LXI:LAN[:SET]:DET event, detect</code><br><code>:TRIG:LXI:LAN[:SET]:DET? event</code>              | Specifies the behavior of the trigger signal.<br><i>event</i> : LAN event name<br><i>detect</i> =RISE FALL HIGH LOW   | RISE            |

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :TRIG:LXI:LAN[:SET]:ENAB <i>event, status</i><br>:TRIG:LXI:LAN[:SET]:ENAB? <i>event</i> | Enables or disables the specified LAN trigger.<br><i>event</i> : LAN event name<br><i>status</i> =0 OFF 1 ON   | OFF           |
| :TRIG:LXI:LAN[:SET]:FILT <i>event, filter</i><br>:TRIG:LXI:LAN[:SET]:FILT? <i>event</i> | Allows user to create a filter expression for the specified LAN trigger event.<br><i>event</i> : LAN event name<br><i>filter</i> : Filter                  | "ALL:5044"    |
| :TRIG:LXI:LAN[:SET]:IDEN <i>event, id</i><br>:TRIG:LXI:LAN[:SET]:IDEN? <i>event</i>     | Sets the string that is expected to arrive over the LAN for a given trigger LAN event to occur.<br><i>event</i> : LAN event name<br><i>id</i> : Identifier | <i>event</i>  |

## Reading Source/Measure Data

Figure 2-11

Measurement Data Flow





**Figure 2-12      Composite Limit Test Flowchart Example for Sorting Mode**

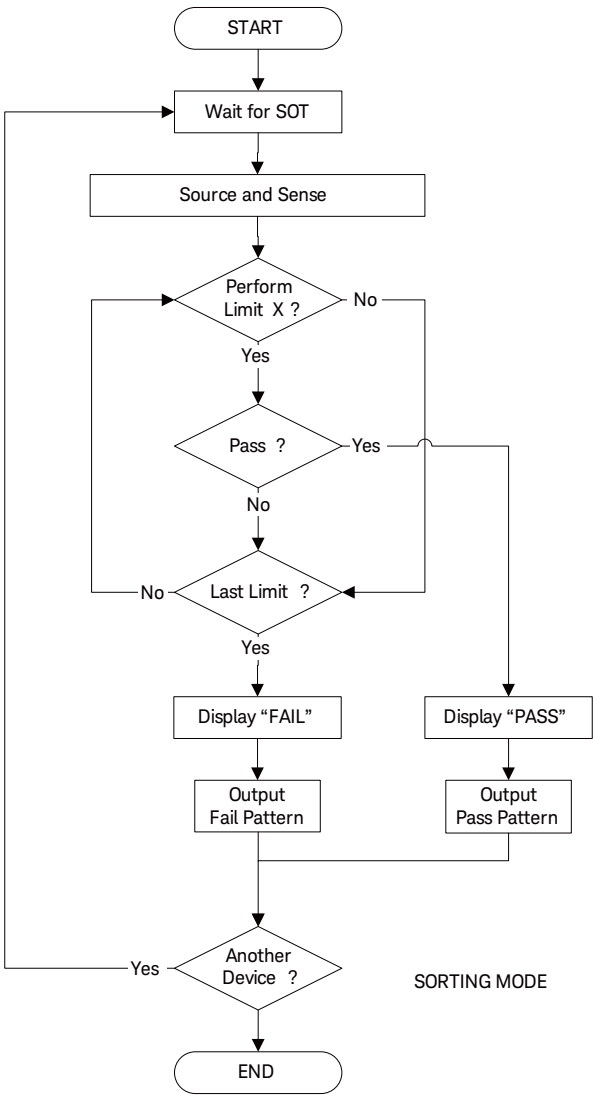
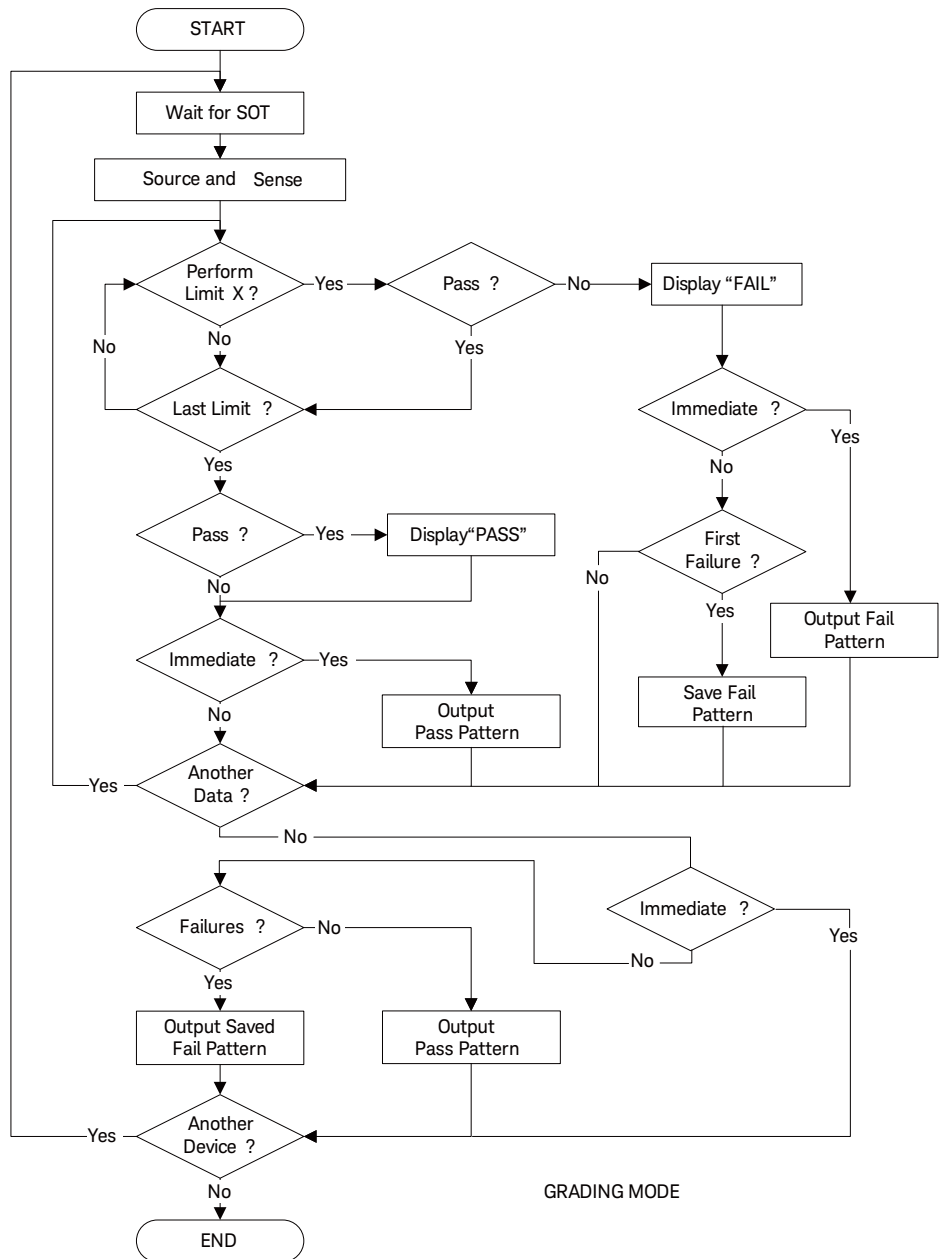


Figure 2-13

Composite Limit Test Flowchart Example for Grading Mode



**Table 2-12 FETCh Subsystem**

| Command   | Summary   | Reset setting |
|---|---|---------------|
| :FETC:ARR? [ <i>chanlist</i> ]  | 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.<br><br><i>chanlist</i> =( <i>@1</i> )( <i>@2</i> )( <i>@1,2</i> )( <i>@1:2</i> )( <i>@2,1</i> )( <i>@2:1</i> )          |               |
| :FETC:ARR:CURR? [ <i>chanlist</i> ]<br>:FETC:ARR:RES? [ <i>chanlist</i> ]<br>:FETC:ARR:SOUR? [ <i>chanlist</i> ]<br>:FETC:ARR:STAT? [ <i>chanlist</i> ]<br>:FETC:ARR:TIME? [ <i>chanlist</i> ]<br>:FETC:ARR:VOLT? [ <i>chanlist</i> ]                   | 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.<br><br><i>chanlist</i> =( <i>@1</i> )( <i>@2</i> )( <i>@1,2</i> )( <i>@1:2</i> )( <i>@2,1</i> )( <i>@2:1</i> ) |               |
| :FETC[:SCAL]? [ <i>chanlist</i> ]   | 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.<br><br><i>chanlist</i> =( <i>@1</i> )( <i>@2</i> )( <i>@1,2</i> )( <i>@1:2</i> )( <i>@2,1</i> )( <i>@2:1</i> )  |               |
| :FETC[:SCAL]:CURR? [ <i>chanlist</i> ]<br>:FETC[:SCAL]:RES? [ <i>chanlist</i> ]<br>:FETC[:SCAL]:SOUR? [ <i>chanlist</i> ]<br>:FETC[:SCAL]:STAT? [ <i>chanlist</i> ]<br>:FETC[:SCAL]:TIME? [ <i>chanlist</i> ]<br>:FETC[:SCAL]:VOLT? [ <i>chanlist</i> ] | 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.<br><br><i>chanlist</i> =( <i>@1</i> )( <i>@2</i> )( <i>@1,2</i> )( <i>@1:2</i> )( <i>@2,1</i> )( <i>@2:1</i> )                               |               |

Subsystem Command Summary  
Reading Source/Measure Data

**Table 2-13 FORMat Subsystem**

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

**Table 2-14 READ Subsystem**

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

Subsystem Command Summary  
Reading Source/Measure Data

**Table 2-15 MEASure Subsystem**

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

**Table 2-16 CALCulate Subsystem**

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

| Command  | Summary  | Reset setting |
|--|--|---------------|
| :CALC[c]:CLIM:FAIL:DIG[:DATA] <i>pattern</i><br>:CALC[c]:CLIM:FAIL:DIG[:DATA]? | Defines a fail pattern that appears near the end of the flowcharts shown in Figures 2-12 and 2-13. This is a bit pattern used to indicate the composite limit test <i>fail</i> .<br><i>pattern</i> =0 to 16383 | 0             |
| :CALC[c]:CLIM:MODE <i>mode</i><br>:CALC[c]:CLIM:MODE?                          | Sets the operation mode of the composite limit test.<br><i>mode</i> =GRADing SORTing.  | GRAD          |
| :CALC[c]:CLIM:PASS:DIG[:DATA] <i>pattern</i><br>:CALC[c]:CLIM:PASS:DIG[:DATA]? | Defines a pass pattern that appears near the end of the flowcharts shown in Figures 2-12 and 2-13. This is a bit pattern used to indicate the composite limit test <i>pass</i> .<br><i>pattern</i> =0 to 16383 | 0             |
| :CALC[c]:CLIM:STAT <i>mode</i><br>:CALC[c]:CLIM:STAT?                          | Enables or disables the composite limit test.<br><i>mode</i> =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 2-12 and 2-13.  |               |
| :CALC[c]:CLIM:UPD <i>result</i><br>: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 2-13.<br><i>result</i> =END IMMediate                                      | IMM           |

Subsystem Command Summary  
Reading Source/Measure Data

| Command  | Summary  | Reset setting     |
|--|--|-------------------|
| :CALC[c]:DATA? [ <i>offset</i> [, <i>size</i> ]]   | :CALC[c]:DATA? returns limit test data.<br><i>offset</i> =CURRENT START 0 to maximum<br><i>size</i> =1 to maximum  | STAR and all data |
| :CALC[c]:DATA:LAT?                                 | :CALC[c]:DATA:LAT? returns the last limit test data.<br><br>Elements of the returned data are specified by the :FORM:ELEM:CALC command. The limit test data can be expressed by the following formula.<br><br><i>limit test data</i> = <i>input data</i> - <i>null offset</i>                  |                   |
| :CALC[c]:DIG:BIT <i>pin</i><br>:CALC[c]:DIG:BIT?   | Assigns the GPIO pins used for the result output. The output is the pass/fail bit pattern.<br><br><i>pin</i> =EXT <i>n</i>  NONE. <i>n</i> =1 to 14.   | NONE              |
| :CALC[c]:DIG:BUSY <i>pin</i><br>:CALC[c]:DIG:BUSY? | Assigns the GPIO pin for the BUSY (busy) line for the composite limit test.<br><br><i>pin</i> =EXT <i>n</i>  NONE. <i>n</i> =1 to 14.  | NONE              |
| :CALC[c]:DIG:EOT <i>pin</i><br>:CALC[c]:DIG:EOT?   | Assigns the GPIO pin for the EOT (end of test) line for the composite limit test.<br><br><i>pin</i> =EXT <i>n</i>  NONE. <i>n</i> =1 to 14.  | NONE              |
| :CALC[c]:DIG:SOT <i>pin</i><br>:CALC[c]:DIG:SOT?   | Assigns the GPIO pin for the SOT (start of test) line for the composite limit test.<br><br><i>pin</i> =EXT <i>n</i>  NONE. <i>n</i> =1 to 14.  | NONE              |
| :CALC[c]:FEED <i>type</i><br>:CALC[c]:FEED?        | Specifies the <i>input data</i> 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 2-12 and 2-13.<br><br><i>type</i> =MATH RESistance CURRENT VOLTage | VOLT              |



| Command  | Summary   | Reset setting |
|--|---|---------------|
| :CALC[c]:LIM[m]:COMP:DIG[:DATA] <i>pattern</i><br>:CALC[c]:LIM[m]:COMP:DIG[:DATA] ?          | Defines the bit pattern used to indicate a <i>failure</i> of the compliance status check specified by <i>m</i> .<br><br><i>pattern</i> =0 to 16383  | 0             |
| :CALC[c]:LIM[m]:COMP:FAIL <i>criter</i><br><i>ia</i><br>:CALC[c]:LIM[m]:COMP:FAIL?           | Sets the judgement criteria for the compliance status check specified by <i>m</i> .<br><br><i>criteria</i> =IN determines that the limit test has failed if the channel goes into the compliance state.<br><br><i>criteria</i> =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 <i>m</i> . 0: Passed, 1: Failed.  |               |
| :CALC[c]:LIM[m]:FUNC <i>type</i><br>:CALC[c]:LIM[m]:FUNC?                                    | Sets the type of the limit test specified by <i>m</i> .<br><br><i>type</i> =COMP sets the compliance status check which checks if the channel is in the compliance status.<br><br><i>type</i> =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 <i>limit</i><br>:CALC[c]:LIM[m]:LOW? [MINimum MA<br>Ximum DEFault]       | Sets a lower limit used for the limit test specified by <i>m</i> .<br><br><i>limit</i> =MINimum MAXimum DEFault <br>-9.999999E+20 to +9.999999E+20  | -1            |
| :CALC[c]:LIM[m]:LOW:DIG[:DATA] <i>p</i><br><i>attern</i><br>:CALC[c]:LIM[m]:LOW:DIG[:DATA] ? | Defines the bit pattern used to indicate <i>failed-by-exceeding-lower-limit</i> of the limit test specified by <i>m</i> . The bit pattern is used for the GRAD composite limit test.<br><br><i>pattern</i> =0 to 16383  | 0             |

Subsystem Command Summary  
Reading Source/Measure Data

| Command   | Summary   | Reset setting     |
|---|---|-------------------|
| :CALC[c]:LIM[m]:PASS:DIG[:DATA]<br><i>pattern</i><br>:CALC[c]:LIM[m]:PASS:DIG[:DATA]?<br><br><i>pattern</i> =0 to 16383       | Defines the bit pattern used to indicate a <i>pass</i> of the limit test specified by <i>m</i> . The bit pattern is used for the SORT composite limit test.<br><br><i>pattern</i> =0 to 16383   | 0                 |
| :CALC[c]:LIM[m]:STAT <i>mode</i><br>:CALC[c]:LIM[m]:STAT?   | Enables or disables the limit test specified by <i>m</i> .<br><br><i>mode</i> =1 ON 0 OFF   | OFF               |
| :CALC[c]:LIM[m]:UPP <i>limit</i><br>:CALC[c]:LIM[m]:UPP? [MINimum MAXimum DEFAULT]  | Sets a upper limit used for the limit test specified by <i>m</i> .<br><br><i>limit</i> =MINimum MAXimum DEFAULT <br>-9.999999E+20 to +9.999999E+20  | 1                 |
| :CALC[c]:LIM[m]:UPP:DIG[:DATA] <i>p</i><br><i>attern</i><br>:CALC[c]:LIM[m]:UPP:DIG[:DATA]?<br><br><i>pattern</i> =0 to 16383 | Defines a bit pattern used to indicate <i>failed-by-exceeding-upper-limit</i> of the limit test specified by <i>m</i> . The bit pattern is used for the GRAD composite limit test.<br><br><i>pattern</i> =0 to 16383  | 0                 |
| :CALC[c]:MATH:DATA? [ <i>offset</i> [, <i>size</i> ]]   | :CALC[c]:MATH:DATA? returns the calculation result data.<br><br><i>offset</i> =CURRENT START 0 to maximum<br><i>size</i> =1 to maximum<br><br>:CALC[c]:MATH:DATA:LAT? returns the latest calculation result data.<br><br>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. | STAR 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,<br>OFFCOMPO<br>HM,<br>VOLTCOEF,<br>VARALPH<br>A |
| :CALC [c] :MATH [:EXPR] [:DEF] <i>definition</i>                            | Defines a math expression which will be a user-defined math expression.<br>Maximum of 32 math expressions can be defined including the predefined math expressions.<br><i>definition</i> : Up to 256 ASCII characters. | (VOLT *<br>CURR)                                       |
| :CALC [c] :MATH [:EXPR] :DEL:ALL  | Deletes all user-defined math expressions.   |  |
| :CALC [c] :MATH [:EXPR] :DEL [:SEL] <i>name</i>                             | Deletes an user-defined math expression.<br><i>name</i> : Up to 32 ASCII characters.   |  |
| :CALC [c] :MATH [:EXPR] :NAME <i>name</i><br>:CALC [c] :MATH [:EXPR] :NAME? | Selects a math expression used for calculation.<br><i>name</i> : Up to 32 ASCII characters without any control characters, space characters, single and double quotes, and comma.                                      | “POWER”  |
| :CALC [c] :MATH:STAT <i>mode</i><br>:CALC [c] :MATH:STAT?                   | Enables or disables the math expression.<br><i>mode</i> =1 ON 0 OFF  | OFF  |
| :CALC [c] :MATH:UNIT <i>name</i><br>:CALC [c] :MATH:UNIT?                   | Defines the unit name for the math expression.<br><i>name</i> : Up to 32 ASCII characters.   | “W”  |

Subsystem Command Summary  
Reading Source/Measure Data

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

**Table 2-17 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? [ <i>offset</i> [, <i>size</i> ]]      | Returns the data in the trace buffer.<br><br><i>offset</i> =CURRENT START 0 to maximum<br><br><i>size</i> =1 to maximum   |               |
| :TRAC[c]:FEED <i>type</i><br>: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.<br><br><i>type</i> =MATH LIMIT SENSe | SENS          |
| :TRAC[c]:FEED:CONT <i>mode</i><br>:TRAC[c]:FEED:CONT? | Selects the trace buffer control.<br><br><i>mode</i> =NEXT NEVer  | NEV           |

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :TRAC[c]:FREE?  | Returns the available size ( <i>available</i> ) and the total size ( <i>total</i> ) of the trace buffer.<br><br>Response is <i>available,total</i> .   |               |
| :TRAC[c]:POIN <i>points</i><br>: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.<br><br><i>points</i> =MINimum MAXimum DEFault <br>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 <i>operation</i><br>:TRAC[c]:STAT:FORM?              | Selects the statistical operation performed by the :TRAC[c]:STAT:DATA? command.<br><br><i>operation</i> =MINimum MAXimum MEAN SDEViation PKPK  | MEAN          |
| :TRAC[c]:TST:FORM <i>rule</i><br>:TRAC[c]:TST:FORM?                     | Selects the rule for reading the timestamp data in the trace buffer.<br><br><i>rule</i> =DELTA ABSolute  | ABS           |

## Using Advanced Functions

**Table 2-18**      **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 <i>format</i><br>:HCOP:SDUM:FORM? | Sets the format of the image data.<br><i>format</i> =JPG BMP PNG WMF  | JPG           |

**Table 2-19**      **DISPlay Subsystem**

| Command   | Summary  | Reset setting                                 |
|---|--|---|
| :DISP:CSET <i>color</i><br>:DISP:CSET?                          | Selects the color set of the front panel display.<br><i>color</i> =1 (default color set) 2 (alternative color set)   |   |
| :DISP:DIG <i>digits</i><br>:DISP:DIG? [MINimum MAXimum DEFault] | Sets the display resolution of the data displayed on the front panel display.<br><i>digits</i> =4 (3½ digits) 5 (4½ digits) 6 (5½ digits) 7 (6½ digits) MINimum MAXimum DEFAULT. | 7   |
| :DISP:ENAB <i>mode</i><br>:DISP:ENAB?                           | Enables or disables the front panel display under remote operation.<br><i>mode</i> =1 ON 0 OFF   |   |
| :DISP:VIEW <i>mode</i><br>:DISP:VIEW?                           | Sets the display mode, single 1, single 2, dual, graph, or roll.<br><i>mode</i> =SINGle1 SINGle2 DUAL GRAPh ROLL   | SING1 for 1-ch models<br>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 <i>text</i><br>: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.<br><br><i>text</i> : Up to 32 ASCII characters. | ""            |
| :DISP[:WIND[d]]:TEXT:STAT<br>:DISP[:WIND[d]]:TEXT:STAT?             | Shows or hides the text message set by the :DISP[:WIND[d]]:TEXT:DATA command.<br><br><i>mode</i> =1 ON 0 OFF   | OFF           |
| :DISP:ZOOM <i>mode</i><br>:DISP:ZOOM?                               | Enables or disables the zoom function of the front panel display.<br><br><i>mode</i> =1 ON 0 OFF   | OFF           |

**Table 2-20**                      **MMEMory Subsystem**

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

Subsystem Command Summary  
Using Advanced Functions

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



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

**Table 2-21**                      **PROG**ram Subsystem

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

Subsystem Command Summary  
Using Advanced Functions

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :PROG[:SEL]:APP <i>program_code</i>                     | Adds a program code to the end of a program stored in the program memory.<br><br><i>program_code</i> : 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 <i>program_code</i><br>:PROG[:SEL]:DEF? | Defines a program in the program memory by entering the initial program code.<br><br><i>program_code</i> : 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 <i>name</i><br>:PROG[:SEL]:NAME?       | Selects the program for performing the action by the following commands.<br><br>If <i>name</i> does not specify the program stored in the program memory, this command creates a new program with the specified name and selects the program.<br><br>If <i>name</i> specifies an existing program, this command selects the program. |               |

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

**Table 2-22**                      **SYSTem Subsystem**

| Command  | Summary  | Reset setting |
|--|--|---------------|
| :SYST:BEEP[:IMM] <i>frequency,time</i>                                     | Generates a beep sound of the specified frequency and duration.<br><br><i>frequency</i> =55 to 6640 Hz<br><i>time</i> =0.05 to 12.75 seconds   |               |
| :SYST:BEEP:STAT <i>mode</i><br>:SYST:BEEP:STAT?                            | Enables or disables the beeper.<br><br><i>mode</i> =0 OFF 1 ON   |               |
| :SYST:COMM:ENAB <i>mode,interface</i><br>:SYST:COMM:ENAB? <i>interface</i> | 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.<br><br><i>mode</i> =0 OFF 1 ON<br><br><i>interface</i> =GPIB USB LAN SOCKets TELNet VXI11 HISLip WEB |               |

Subsystem Command Summary  
Using Advanced Functions

| Command  | Summary   | Reset setting |
|--|---|---------------|
| :SYST:COMM:GPIB[:SELF]:ADDR <i>address</i><br>:SYST:COMM:GPIB[:SELF]:ADDR? | Sets the GPIB address of the instrument.<br><i>address</i> =0 to 30   |               |
| :SYST:COMM:LAN:ADDR <i>address</i><br>: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.<br><i>address</i> = <i>A.B.C.D</i> , 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 225.<br>CURR: Present setup value<br>STAT: Reserved value for the next startup |               |
| :SYST:COMM:LAN:BST?  | Returns the LAN boot status of the instrument.<br>Response is LAN_AUTO_IP, LAN_DHCP, LAN_FAULT, or LAN_STATIC.  |               |
| :SYST:COMM:LAN:CONT?<br>:SYST:COMM:TCP:CONT?                               | Returns the control connection port number of the specified port.   |               |
| :SYST:COMM:LAN:DHCP <i>mode</i><br>: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.<br><i>mode</i> =0 OFF 1 ON   |               |
| :SYST:COMM:LAN:DNS[j] <i>address</i><br>:SYST:COMM:LAN:DNS[j]? [CURR STAT] | Sets the IP address of the DNS server.<br><i>address</i> = <i>A.B.C.D</i> , 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 255.<br>CURR: Present setup value<br>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 <i>address</i><br>:SYST:COMM:LAN:GATE <i>address</i><br>:SYST:COMM:LAN:GAT? [CURR STAT]<br>: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.<br><br><i>address=A.B.C.D</i> , 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 225.<br><br>CURR: Present setup value<br>STAT: Reserved value for the next startup |               |
| :SYST:COMM:LAN:HNAM <i>hostname</i><br>:SYST:COMM:LAN:HOST <i>hostname</i><br>:SYST:COMM:LAN:HNAM? [CURR STAT]<br>:SYST:COMM:LAN:HOST? [CURR STAT] | Sets the host name of the instrument. The setting is enabled by the :SYST:COMM:LAN:UPD command.<br><br><i>hostname</i> : Host name. Up to 15 characters.<br><br>CURR: Present setup value<br>STAT: Reserved value for the next startup   |               |
| :SYST:COMM:LAN:MAC?  | Returns the MAC address of the instrument.   |               |
| :SYST:COMM:LAN:SMAS <i>subnet_mask</i><br>:SYST:COMM:LAN:SMAS? [CURR STAT]   | Sets the static subnet mask. The setting is enabled by the :SYST:COMM:LAN:UPD command.<br><br><i>subnet_mask=A.B.C.D</i> , 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 255.<br><br>CURR: Present setup value<br>STAT: Reserved value for the next startup            |               |
| :SYST:COMM:LAN:TELN:PROM <i>prompt</i><br>:SYST:COMM:LAN:TELN:PROM?  | Sets the command prompt displayed during a Telnet session for establishing communication with the instrument.<br><br><i>prompt</i> : Command prompt. Up to 15 characters.  |               |

Subsystem Command Summary  
Using Advanced Functions

| Command  | Summary   | Reset setting |
|--|---|---------------|
| :SYST:COMM:LAN:TELN:WMES <i>message</i><br>:SYST:COMM:LAN:TELN:WMES?         | Sets the welcome message displayed during a Telnet session when starting the communication with the instrument.<br><br><i>message</i> : 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[j] <i>address</i><br>:SYST:COMM:LAN:WINS[j]? [CURR STAT] | Sets the IP address of the WINS server.<br><br><i>address</i> = <i>A.B.C.D</i> , 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 255.<br><br>CURR: Present setup value<br>STAT: Reserved value for the next startup |               |
| :SYST:DATA:QUAN? [ <i>chanlist</i> ]   | Returns the number of data for the specified channel in the data buffer.<br><br><i>chanlist</i> =( <i>@1</i> ) ( <i>@2</i> ) ( <i>@1,2</i> ) ( <i>@1:2</i> ) ( <i>@2,1</i> ) ( <i>@2:1</i> )  |               |
| :SYST:DATE <i>year, month, day</i><br>:SYST:DATE?                            | Sets the date of the internal clock.<br><br><i>year</i> : Year. 4-digit integer.<br><br><i>month</i> : Month. Integer from 1 to 12.<br><br><i>day</i> : 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 <i>mode</i><br>:SYST:FAN:MODE?       | Sets the fan control mode.<br><i>mode</i> =NORMal RACK   |               |
| :SYST:GRO[:DEF] <i>chanlist</i><br>:SYST:GRO[:DEF]? | Defines the channel group.<br><i>group</i> list=(@1,2) for making group, or<br><i>group</i> list=(@1),(@2) for breaking group                            |               |
| :SYST:GRO:RES                                       | Releases the channel group.  |               |
| :SYST:INT:TRIP?                                     | Returns if the interlock circuit is close or open.<br><br>Response is 0 or 1 that indicates the interlock circuit is close or open, respectively         |               |
| :SYST:LANG <i>mode</i><br>:SYST:LANG?               | Selects the B2900 control command set.<br>If the setting is changed, the instrument will be automatically rebooted.<br><br><i>mode</i> ="DEFault" "2400" |               |
| :SYST:LFR <i>frequency</i><br>:SYST:LFR?            | Selects the line frequency.<br><br><i>frequency</i> =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.  |               |

Subsystem Command Summary  
Using Advanced Functions

| Command   | Summary   | Reset setting |
|---|---|---------------|
| :SYST:LOCK:REQ?   | Requests a lock of the current I/O interface.   |               |
| :SYST:PERS[:MAN] <i>maker</i><br>:SYST:PERS[:MAN]? [ <i>maker</i> ] | Changes the manufacturer name of the *IDN? command response.<br><br><i>maker</i> =DEF AGIL KEYS or a quoted string<br><br>Query does not support a quoted string.                               |               |
| :SYST:PERS[:MAN]:DEF  | Clears the :SYST:PERS[:MAN] command setting and applies the default value “Keysight Technologies.”  |               |
| :SYST:PERS:MOD <i>model</i><br>:SYST:PERS:MOD?                      | Changes the model number of the *IDN? command response.<br><br><i>model</i> : a quoted string   |               |
| :SYST:PERS:MOD:DEF  | Clears the :SYST:PERS:MOD command setting and applies the default value (the model number on the front panel).  |               |
| :SYST:PON <i>memory</i>   | Specifies the power-on state.<br><br><i>memory</i> =RST RCL0 RCL1 RCL2 RCL3 RCL4  | RST           |
| :SYST:PRES  | Presets the instrument settings and the front panel display.  |               |
| :SYST:SET <i>data</i><br>:SYST:SET?                                 | Sends or loads the instrument setup data.<br><br><i>data</i> : Instrument setup data. Parameter data type is a definite length arbitrary binary block.  |               |
| :SYST:TIME <i>hour,minute,second</i><br>:SYST:TIME?                 | Sets the time of the internal clock.<br><br><i>hour</i> : Hour. Integer from 0 to 23.<br><br><i>minute</i> : Minute. Integer from 0 to 59.<br><br><i>second</i> : Second. Integer from 0 to 59. |               |
| :SYST:TIME:TIM:COUN?  | Returns the present count of the timer.   |               |



| Command   | Summary  | Reset setting |
|---|--|---------------|
| :SYST:TIME:TIM:COUN:RES:AUTO <i>mode</i><br>: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.<br><br><i>mode</i> =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 2-23                      STATus Subsystem**

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :STAT:MEAS:COND?<br>:STAT:OPER:COND?<br>:STAT:QUES:COND?  | Returns the value of the measurement, operation, or questionable status condition register.  |               |
| :STAT:MEAS:ENAB <i>mask</i><br>:STAT:OPER:ENAB <i>mask</i><br>:STAT:QUES:ENAB <i>mask</i><br>:STAT:MEAS:ENAB?<br>:STAT:OPER:ENAB?<br>:STAT:QUES:ENAB? | Sets the measurement, operation, or questionable status enable register.<br><br><i>mask</i> =0 to 65535 (decimal)  | 0             |
| :STAT:MEAS[:EVEN]?<br>:STAT:OPER[:EVEN]?<br>:STAT:QUES[:EVEN]?  | Returns the value of the measurement, operation, or questionable status event register.  |               |
| :STAT:MEAS:NTR <i>filter</i><br>:STAT:OPER:NTR <i>filter</i><br>:STAT:QUES:NTR <i>filter</i><br>:STAT:MEAS:NTR?<br>:STAT:OPER:NTR?<br>: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.<br><br><i>filter</i> =0 to 65535 (decimal) | 0             |

Subsystem Command Summary  
Using Advanced Functions

| Command   | Summary  | Reset setting |
|---|--|---------------|
| :STAT:MEAS:PTR <i>filter</i><br>:STAT:OPER:PTR <i>filter</i><br>:STAT:QUES:PTR <i>filter</i><br>:STAT:MEAS:PTR?<br>:STAT:OPER:PTR?<br>: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.<br><br><i>filter</i> =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.   |               |
| :STAT:QUES:CAL:COND?<br>:STAT:QUES:CURR:COND?<br>:STAT:QUES:TEMP:COND?<br>:STAT:QUES:TEST:COND?<br>:STAT:QUES:VOLT:COND?  | Returns the value of the questionable status condition register.   |               |
| :STAT:QUES:CAL:ENAB <i>mask</i><br>:STAT:QUES:CURR:ENAB <i>mask</i><br>:STAT:QUES:TEMP:ENAB <i>mask</i><br>:STAT:QUES:TEST:ENAB <i>mask</i><br>:STAT:QUES:VOLT:ENAB <i>mask</i><br>:STAT:QUES:CAL:ENAB?<br>:STAT:QUES:CURR:ENAB?<br>:STAT:QUES:TEMP:ENAB?<br>:STAT:QUES:TEST:ENAB?<br>:STAT:QUES:VOLT:ENAB? | Sets the questionable calibration, current, temperature, test, or voltage status enable register.<br><br><i>mask</i> =0 to 65535 (decimal)   | 0             |

| Command   | Summary   | Reset setting |
|---|---|---------------|
| :STAT:QUES:CAL[:EVEN]?<br>:STAT:QUES:CURR[:EVEN]?<br>:STAT:QUES:TEMP[:EVEN]?<br>:STAT:QUES:TEST[:EVEN]?<br>:STAT:QUES:VOLT[:EVEN]?  | Returns the value of the questionable calibration, current, temperature, test, or voltage status event register.  |               |
| :STAT:QUES:CAL:NTR <i>filter</i><br>:STAT:QUES:CURR:NTR <i>filter</i><br>:STAT:QUES:TEMP:NTR <i>filter</i><br>:STAT:QUES:TEST:NTR <i>filter</i><br>:STAT:QUES:VOLT:NTR <i>filter</i><br>:STAT:QUES:CAL:NTR?<br>:STAT:QUES:CURR:NTR?<br>:STAT:QUES:TEMP:NTR?<br>:STAT:QUES:TEST:NTR?<br>: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.<br><br><i>filter</i> =0 to 65535 (decimal) | 0             |
| :STAT:QUES:CAL:PTR <i>filter</i><br>:STAT:QUES:CURR:PTR <i>filter</i><br>:STAT:QUES:TEMP:PTR <i>filter</i><br>:STAT:QUES:TEST:PTR <i>filter</i><br>:STAT:QUES:VOLT:PTR <i>filter</i><br>:STAT:QUES:CAL:PTR?<br>:STAT:QUES:CURR:PTR?<br>:STAT:QUES:TEMP:PTR?<br>:STAT:QUES:TEST:PTR?<br>: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.<br><br><i>filter</i> =0 to 65535 (decimal) | 32767         |

Subsystem Command Summary  
Using Advanced Functions



This chapter describes common commands and queries of *IEEE 488.2*. The commands available for Keysight 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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | <i>*ESE enable_number</i><br><i>*ESE?</i>  |   |
| <b>Parameter</b>      | <i>enable_number</i>   | 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. |
| <b>Query response</b> | <i>enable_number</i> <newline><^END><br><br><i>enable_number</i> is the sum of the binary-weighted values of the Enable register bits. The return format can be selected by the <b>:FORMat:SREGister</b> command. Response data type is NR1 or NDN.  |   |
| <b>Remarks</b>        | Bit definitions of the Standard Event register are shown in <a href="#">Table 3-2</a> . 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.<br><br>The <b>*CLS</b> (clear status) command will not clear the enable register but it does clear all bits in the event register.<br><br>The <b>:STATus:PRESet</b> command does not clear the bits in the Status Byte register. |   |
| <b>See Also</b>       | <b>:FORMat:SREGister</b> and <b>:STATus:PRESet</b>   |   |

---

## **\*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 <a href="#">Chapter 5, “Error Messages.”</a> |
| 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

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

*model*: mainframe model number

*serial*: mainframe serial number

*revision*: firmware revision number

Response data type is AARD.

### Remarks

The query response can be changed by using the following commands. For details, see each command description.

- **:SYSTem:PERSon[:MANUFACTURer]**
- **:SYSTem:PERSon[:MANUFACTURer]:DEFault**
- **:SYSTem:PERSon:MODEl**
- **:SYSTem:PERSon:MODEl:DEFault**

---

## \*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 QUESTionabl e>: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 QUESTionabl e>: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 QUESTionabl e>: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 Keysight 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**



This chapter describes subsystem commands available for Keysight 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 4-182.

*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 2-12 and 2-13. 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 2-13](#) 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 2-12](#) 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 2-12](#) and [2-13](#).

**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>

Subsystem Commands  
CALCulate Subsystem

*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 2-12 and 2-13.

**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 2-13.

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*]]

|                  |                      |  |
|------------------|----------------------|--|
| <b>Parameter</b> | <b><i>offset</i></b> | Indicates the beginning of the data received. <i>n</i>  CURRent STARt (default). Parameter data type is NR1 or CPD.<br><br><i>offset</i> = <i>n</i> specifies the <i>n</i> +1th data. <i>n</i> is an integer, 0 to maximum (depends on the buffer state).<br><br><i>offset</i> =CURR specifies the present data position.<br><br><i>offset</i> =STAR specifies the top of the data buffer. Same as <i>offset</i> =0. |
|                  | <b><i>size</i></b>   | 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 <i>offset</i> 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 2-12 and 2-13. The limit test data is returned by the :CALCulate:DATA? or :CALCulate:DATA:LATest? command.

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

## Subsystem Commands

### CALCulate Subsystem

:CALCulate[*c*]:FEED?

|                  |  |
|------------------|--|
| <b>Parameter</b> | <p><b><i>type</i></b> Data type. MATH RESistance CURRent VOLTage (default). Parameter data type is CPD.</p> <p><i>type</i>=VOLT specifies the voltage measurement data.</p> <p><i>type</i>=CURR specifies the current measurement data.</p> <p><i>type</i>=RES specifies the resistance calculation data given by the following formula.</p> $\text{Resistance} = V_{\text{meas}} / I_{\text{meas}}$ <p>Where, <i>Vmeas</i> is the voltage measurement data, and <i>Imeas</i> is the current measurement data.</p> <p><i>type</i>=MATH specifies the data given by a math expression. The math expression must be specified before the :CALC:FEED MATH command is executed.</p> <p>An existing math expression can be specified by the :CALCulate:MATH[:EXPRession]:NAME command.</p> <p>A new math expression can be defined by the :CALCulate:MATH[:EXPRession]:NAME and :CALCulate:MATH[:EXPRession][:DEFine] commands.</p> |
|------------------|--|

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <p><i>type</i> &lt;newline&gt;</p> <p><i>type</i> returns the present setting of data type, MATH, RES, CURR, or VOLT. Response data type is CRD.</p> |
|-----------------------|--|

|                |   |
|----------------|---|
| <b>Example</b> | <pre>:CALC:FEED MATH :CALC2:FEED?</pre> |
|----------------|---|

## :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.

|               |  |
|---------------|--|
| <b>Syntax</b> | <pre>:CALCulate[<i>c</i>]:LIMit[<i>m</i>]:COMPLiance:DIGital[:DATA] <i>bit_pattern</i> :CALCulate[<i>c</i>]:LIMit[<i>m</i>]:COMPLiance:DIGital[:DATA]?</pre> |
|---------------|--|

|                  |   |
|------------------|---|
| <b>Parameter</b> | <p><b><i>bit_pattern</i></b> 0 (default setting in decimal expression) to 16383. Parameter data type is NR1 or NDN.</p> |
|------------------|---|



**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*]]

|                  |                      |   |
|------------------|----------------------|---|
| <b>Parameter</b> | <b><i>offset</i></b> | Indicates the beginning of the data received. <i>n</i>  CURRENT START (default). Parameter data type is NR1 or CPD.<br><br><i>offset=n</i> specifies the <i>n</i> +1th data. <i>n</i> is an integer, 0 to maximum (depends on the buffer state).<br><br><i>offset=CURR</i> specifies the present data position.<br><br><i>offset=STAR</i> specifies the top of the data buffer. Same as <i>offset=0</i> . |
|                  | <b><i>size</i></b>   | 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 <i>offset</i> 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 <sup>a</sup> |           | 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       |

- 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             | ( )      | Parentheses                                   |
| ⋮                | + and -  | Unary plus operator and unary minus operator  |
| ⋮                | ^        | Exponentiation operator                       |
| ⋮                | * and /  | Multiplication operator and division operator |
| Low              | + 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.

$$\text{POWER} = \text{VOLT}[c] * \text{CURR}[c]$$

**OFFCOMPOHM**

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

$$\text{OFFCOMPOHM} = (\text{VOLT}[c][1] - \text{VOLT}[c][0]) / (\text{CURR}[c][1] - \text{CURR}[c][0])$$

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.

$$\text{VARALPHA} = \log(\text{CURR}[c][1] / \text{CURR}[c][0]) / \log(\text{VOLT}[c][1] / \text{VOLT}[c][0])$$

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.



## VOLTCOE

Calculates voltage coefficient using the following formula.

$$\text{VOLTCOE} = (\text{RES}[c][1] - \text{RES}[c][0]) / (\text{RES}[c][1] * (\text{VOLT}[c][1] - \text{VOLT}[c][0])) * 100 \%$$

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.

## Subsystem Commands

### CALCulate Subsystem

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.

|                       |   |   |
|-----------------------|---|---|
| <b>Syntax</b>         | :CALCulate[ <i>c</i> ]:MATH:UNITs <i>unit</i><br>:CALCulate[ <i>c</i> ]:MATH:UNITs?                           |   |
| <b>Parameter</b>      | <i>unit</i>   | Unit name. Up to 32 ASCII characters. Parameter data type is SPD. |
| <b>Query response</b> | <i>unit</i> <newline><br><i>unit</i> returns the unit name of the math expression. Response data type is SRD. |   |
| <b>Example</b>        | :CALC:MATH:UNIT "amps"<br>: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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | :CALCulate[ <i>c</i> ]:OFFSet <i>offset</i><br>:CALCulate[ <i>c</i> ]:OFFSet? [ <i>offset</i> ]  |   |
| <b>Parameter</b>      | <i>offset</i>  | <i>value</i> (-9.999999E+20 to +9.999999E+20) MINimum MAXimum DEFault (default is 0.0). Parameter data type is NRf+. Query does not support <i>offset=value</i> . |
| <b>Query response</b> | <i>offset</i> <newline><br><i>offset</i> returns the present setting of the null offset value. If a parameter is specified, <i>offset</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3. |   |
| <b>Example</b>        | :CALC:OFFS 0.5<br>: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.

|                       |  |  |
|-----------------------|--|--|
| <b>Syntax</b>         | :DISPlay:CSET <i>color</i><br>:DISPlay:CSET? |  |
| <b>Parameter</b>      | <i>color</i>                                 | Color set of the front panel display. 1 2. Parameter data type is NR1.<br><br><i>color</i> =1 selects the default color set.<br><i>color</i> =2 selects the alternative color set. |
| <b>Query response</b> | <i>response</i> <newline>                    | <i>response</i> is 1 or 2, and indicates the color set of the front panel display. Response data type is NR1.  |
| <b>Example</b>        | :DISP:CSET 1<br>:DISP:CSET?                  |  |

### :DISPlay:DIGits

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

|                  |   |   |
|------------------|---|---|
| <b>Syntax</b>    | :DISPlay:DIGits <i>digits</i><br>:DISPlay:DIGits? [MINimum MAXimum DEFault] |   |
| <b>Parameter</b> | <i>digits</i>   | Resolution. <i>value</i> (4 to 7) MINimum MAXimum DEFault (default is 7). Parameter data type is NRf+. Query does not support <i>digits=value</i> .<br><br><i>digits</i> =4 selects 3½ digit resolution.<br><i>digits</i> =5 selects 4½ digit resolution. |

Subsystem Commands  
DISPlay Subsystem

*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  $\mu$  and  $\Omega$  are converted to u and ohm, respectively.

## Subsystem Commands

### DISPlay Subsystem

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,+9.910  
000E+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.  
ASCI|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*

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | :FORMat:ELEMents:CALCulate <i>type</i> {, <i>type</i> }<br>:FORMat:ELEMents:CALCulate?   |   |
| <b>Parameter</b>      | <i>type</i>  | Data element included in the data. CALC (calculation data, default) TIME STATus. Parameter data type is CPD.<br><br>CALC selects the calculation data <i>calc</i> .<br>TIME selects the time (timestamp) data <i>time</i> .<br>STAT selects the status data <i>status</i> . |
| <b>Query response</b> | <i>type</i> {, <i>type</i> } <newline><br><i>type</i> returns CALC, TIME, or STAT. Response data type is CRD. Multiple responses are separated by a comma. |   |
| <b>Example</b>        | :FORM:ELEM:CALC CALC,TIME,STAT<br>: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

### :ARM:LXI:COUNt

B2900 ignores this command. Specifies an integer as the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the measurement must be initiated again. *intRepetitions* must be greater than zero (0) and is limited by either your application or the maximum 32 bit signed integer value. This is not enforced by LxiMiddleware; your SCPI action for this command will need to enforce the limits.

---

#### NOTE

The LXI Middleware only stores this value. In other words, you essentially have to implement the count (e.g., you need to keep track of how many ARM events have occurred, and only arm the instrument when the total COUNt has occurred).

---

#### Syntax

:ARM:LXI:COUNt *intRepetitions*

:ARM:LXI:COUNt?

#### Parameter

*intRepetitions*      Number of arms required to complete the arm loop. Parameter data type is NRf.

#### Query response

*intRepetitions* <newline>

*intRepetitions* returns the number of arms required to complete the arm loop. Response data type is NR1.

#### Example

:ARM:LXI:COUN 10

:ARM:LXI:COUN?

## :ARM:LXI:DElay

Specifies the delay, in seconds, from when the arm logic satisfied until the waiting for trigger state is entered.

---

### NOTE

Any <delay> value other than 0.0 is changed to 0 (zero) and a warning is inserted into the SCPI error queue.

This implementation assumes that there is no Precision Time Protocol (PTP) implementation on the instrument. Delay requires PTP, so without it the delay must be zero.

---

### Syntax

:ARM:LXI:DElay *delay*

:ARM:LXI:DElay?

### Parameter

*delay* Delay, in seconds. Parameter data type is NRf.

### Query response

*delay* <newline>

*delay* returns the delay, in seconds. Response data type is NR3.

### Example

:ARM:LXI:DEL 10.0

:ARM:LXI:DEL?

## :ARM:LXI:LAN[:SET]:DETection

Specifies the style of arm source detection for the specified event. If the source is a LAN event and the source detection is set to rise, this Arm repeated capability will be satisfied when the designated LAN packet arrives with a True indication. If the source detection is set to fall, this Arm repeated capability will be satisfied when a LAN packet arrives with a False indication. If the detection is set to high, the source will be satisfied when the designated LAN packet arrives with a True indication and remain satisfied until the designated LAN packet arrives with a False indication. If the detection is to low, the source will be satisfied when the designated LAN packet arrives with a False indication and remain satisfied until the designated LAN packet arrives with a True indication.

### Syntax

:ARM:LXI:LAN[:SET]:DETection *strLanEvent*, *detection*

:ARM:LXI:LAN[:SET]:DETection? *strLanEvent*

|                  |                           |  |
|------------------|---------------------------|--|
| <b>Parameter</b> | <b><i>strLanEvent</i></b> | LAN event name. String enclosed in quotes. Parameter data type is SPD.   |
|                  | <b><i>detection</i></b>   | Detection type. RISE (default) FALL HIGH LOW. Parameter data type is CPD.  |
|                  |                           | RISE sets the instrument to trigger on the receipt of a signal LOW LAN event followed by a signal HIGH LAN event (rising edge).  |
|                  |                           | FALL sets the instrument to trigger on the receipt of a signal HIGH LAN event followed by a signal LOW LAN event (falling edge). |
|                  |                           | HIGH sets the instrument to trigger on every signal HIGH LAN event.  |
|                  |                           | LOW sets the instrument to trigger on every signal LOW LAN event.  |

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <i>detection</i> <newline>   |
|                       | <i>detection</i> returns a string with the DETection setting for the specified event. Response data type is CRD. |

|                |                              |
|----------------|------------------------------|
| <b>Example</b> | :ARM:LXI:LAN:DET "LAN0",RISE |
|                | :ARM:LXI:LAN:DET? "LAN0"     |

## **:ARM:LXI:LAN[:SET]:ENABle**

Enables or disables the arm source of the specified event.

|               |  |
|---------------|--|
| <b>Syntax</b> | :ARM:LXI:LAN[:SET]:ENABle <i>strLanEvent</i> , <i>status</i> |
|               | :ARM:LXI:LAN[:SET]:ENABle? <i>strLanEvent</i>                |

|                  |                           |  |
|------------------|---------------------------|--|
| <b>Parameter</b> | <b><i>strLanEvent</i></b> | LAN event name. String enclosed in quotes. Parameter data type is SPD. |
|                  | <b><i>status</i></b>      | Status. 0 OFF (default) 1 ON. Parameter data type is boolean.          |

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <i>status</i> <newline>  |
|                       | <i>status</i> returns an integer (0 or 1) denoting the enabled status of the specified event. Response data type is boolean. |

|                |                            |
|----------------|----------------------------|
| <b>Example</b> | :ARM:LXI:LAN:ENAB "LAN0",1 |
|                | :ARM:LXI:LAN:ENAB? "LAN0"  |

## :ARM:LXI:LAN[:SET]:FILTer

Specifies a filter for restricting arm sources of the specified event.

### Syntax

:ARM:LXI:LAN[:SET]:FILTer *strLanEvent*, *strFilterExpression*

:ARM:LXI:LAN[:SET]:FILTer? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***strFilterExpression*** Filter. It will be an existing LAN arm sources, e.g. one of the items returned by :LXI:EVENT:INPut:LAN:LIST? or :LXI:EVENT[:OUTPut]:LAN:LIST?. String enclosed in quotes. Parameter data type is SPD.

### Query response

*strFilterExpression* <newline>

*strFilterExpression* returns a string with the filter expression for the specified event. Response data type is SRD.

### Example

:ARM:LXI:LAN:FILT "LAN0","ALL:5024"

:ARM:LXI:LAN:FILT? "LAN0"

## :ARM:LXI:LAN[:SET]:IDENtifier

Specifies the LAN event identifier that is associated with this arm source. LAN Events with this identifier are accepted from the source described in the filter.

### Syntax

:ARM:LXI:LAN[:SET]:IDENtifier *strLanEvent*, *strCustomId*

:ARM:LXI:LAN[:SET]:IDENtifier? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***strCustomId*** Identifier. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

### Query response

*strCustomId* <newline>

*strCustomId* returns the identifier string for the specified event. Response data type is SRD.

**Example** :ARM:LXI:LAN:IDEN “LAN0”,”MyEvent”  
:ARM:LXI:LAN:IDEN? “LAN0”

## **:LXI:EVENT:DOMain**

Specifies the LXI LAN domain. <bytDomain> is an integer, 0 to 255. A domain facilitates multiple systems on a single LAN. LXI modules ignore all LXI messages except those in its own domain, as defined by the domain byte.

**Syntax** :LXI:EVENT:DOMain *bytDomain*  
:LXI:EVENT:DOMain?

**Parameter** *bytDomain* LXI LAN domain. 0 (default) to 255. Parameter data type is NRf.

**Query response** *bytDomain* <newline>  
*bytDomain* returns the LXI LAN domain. Response data type is NR1.

**Example** :LXI:EVENT:DOM 1  
:LXI:EVENT:DOM?

## **:LXI:EVENT:INPut:LAN:ADD**

B2900 ignores this command. Creates a new input event to receive.

**Syntax** :LXI:EVENT:INPut:LAN:ADD *strLanEvent*

**Parameter** *strLanEvent* LAN event name. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like “@”, “%”, “\*”, etc. Parameter data type is SPD.

**Example** :LXI:EVENT:INP:LAN:ADD “MyEvent”

## **:LXI:EVENT:INPut:LAN:COUNT?**

Returns an integer as the total number of defined input LAN events (includes both enabled and disabled events).

**Syntax** :LXI:EVENT:INPut:LAN:COUNT?



**Query response**     *count* <newline>  
*count* returns an integer as the total number of defined input LAN events. Response data type is NR1.

**Example**             :LXI:EVEN:INP:LAN:COUN?

### **:LXI:EVENT:INPut:LAN:DISable:ALL**

Disables all input events.

**Syntax**             :LXI:EVENT:INPut:LAN:DISable:ALL

**Example**             :LXI:EVEN:INP:LAN:DIS:ALL

### **:LXI:EVENT:INPut:LAN:LIST?**

Returns a quoted string with the list of defined input event names.

**Syntax**             :LXI:EVENT:INPut:LAN:LIST?

**Query response**     *list* <newline>  
*list* returns a quoted string with the list of defined input event names. Response data type is SRD.

**Example**             :LXI:EVEN:INP:LAN:LIST?

### **:LXI:EVENT:INPut:LAN:REMove:ALL**

B2900 ignores this command. Removes all of the input events that were added using **:LXI:EVENT:INPut:LAN:ADD**. Predefined LAN names (LAN0 - LAN7) cannot be removed.

**Syntax**             :LXI:EVENT:INPut:LAN:REMove:ALL

**Example**             :LXI:EVEN:INP:LAN:REM:ALL

## :LXI:EVENT:INPut:LAN:REMove

B2900 ignores this command. Removes the specified input event from the list of named events. Predefined LAN event names (LAN0 - LAN7) cannot be removed.

The event is added by :LXI:EVENT:INPut:LAN:ADD.

**Syntax** :LXI:EVENT:INPut:LAN:REMove *strLanEvent*

**Parameter** *strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

**Example** :LXI:EVENT:INP:LAN:REM "MyEvent"

## :LXI:EVENT:INPut:LAN[:SET]:CONFigure

Configures the most common attributes of LXI LAN input events. The parameters are also available from their corresponding commands. For more details on the configurable parameters, refer to the individual SCPI commands below.

**Syntax** :LXI:EVENT:INPut:LAN[:SET]:CONFigure *strLanEvent*, *enable*, *detection* [, *delay*], *filter*, *identifier*

**Parameter** *strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

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

*detection* Detection type. RISE|FALL|HIGH|LOW. Parameter data type is CPD.

*delay* Delay, in seconds. Parameter data type is NRf.

*filter* Filter. String enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

*identifier* Identifier. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

RISE sets the instrument to trigger on the receipt of a signal LOW LAN event followed by a signal HIGH LAN event (rising edge).

FALL sets the instrument to trigger on the receipt of a signal HIGH LAN event followed by a signal LOW LAN event (falling edge).

HIGH sets the instrument to trigger on every signal HIGH LAN event.

LOW sets the instrument to trigger on every signal LOW LAN event.

**Example** :LXI:EVEN:INP:LAN:CONF "LAN0",ON,"HIGH",0,"ALL:5044","LAN0"

## :LXI:EVENT:INPut:LAN[:SET]:DElay

The delay is optional. <delay> is a double representing the delay, in seconds. Any <delay> value other than 0.0 is changed to 0 (zero) and a warning is inserted into the SCPI error queue.

### NOTE

This implementation assumes that there is no Precision Time Protocol (PTP) implementation on the instrument. Delay requires PTP, so without it the delay must be zero.

### Syntax

:LXI:EVENT:INPut:LAN[:SET]:DElay *strLanEvent*, *delay*

:LXI:EVENT:INPut:LAN[:SET]:DElay? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***delay*** Delay, in seconds. Parameter data type is NRf.

### Query response

*delay* <newline>

*delay* returns the delay, in seconds, for the specified LAN trigger event. Response data type is NR3.

### Example

:LXI:EVEN:INP:LAN:DEL "LAN0",10.0

:LXI:EVEN:INP:LAN:DEL? "LAN0"

## :LXI:EVENT:INPut:LAN[:SET]:DETection

Specifies the trigger detection method and polarity for the input event.

### Syntax

:LXI:EVENT:INPut:LAN[:SET]:DETection *strLanEvent*, *detection*

:LXI:EVENT:INPut:LAN[:SET]:DETection? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***detection*** Detection type. RISE (default)|FALL|HIGH|LOW. Parameter data type is CPD.

RISE sets the instrument to trigger on the receipt of a signal LOW LAN event followed by a signal HIGH LAN event (rising edge).

FALL sets the instrument to trigger on the receipt of a signal HIGH LAN event followed by a signal LOW LAN event (falling edge).

HIGH sets the instrument to trigger on every signal HIGH LAN event.

LOW sets the instrument to trigger on every signal LOW LAN event.

**Query response** *detection* <newline>  
*detection* returns a character string with the detection setting. Response data type is CRD.

**Example**  
:LXI:EVEN:INP:LAN:DET "LAN0",RISE  
:LXI:EVEN:INP:LAN:DET? "LAN0"

## **:LXI:EVENT:INPut:LAN[:SET]:ENABLE**

Enables or disables the specified input event. The event is ignored if disabled.

**Syntax**  
:LXI:EVENT:INPut:LAN[:SET]:ENABLE *strLanEvent*, *status*  
:LXI:EVENT:INPut:LAN[:SET]:ENABLE? *strLanEvent*

**Parameter**  
***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.  
***status*** Status. 0|OFF (default)|1|ON. Parameter data type is boolean.

**Query response** *status* <newline>  
*status* returns an integer (0 or 1) denoting the enabled status of the specified event. Response data type is boolean.

**Example**  
:LXI:EVEN:INP:LAN:ENAB "LAN0",1  
:LXI:EVEN:INP:LAN:ENAB? "LAN0"

## :LXI:EVENT:INPut:LAN[:SET]:FILTer

Creates a filter for incoming input events. Only input events coming from hosts matching the filter string are processed.

### Syntax

:LXI:EVENT:INPut:LAN[:SET]:FILTer *strLanEvent*, *strFilterExpression*

:LXI:EVENT:INPut:LAN[:SET]:FILTer? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

### ***strFilterExpression***

Filter which specifies an existing LAN event, e.g. one of the items returned by :LXI:EVENT:INPut:LAN:LIST?. String enclosed in quotes. Parameter data type is SPD.

### Query response

*strFilterExpression* <newline>

*strFilterExpression* returns a string with the filter expression for the specified event. Response data type is SRD.

### Example

:LXI:EVENT:INP:LAN:FILT "LAN0",1

:LXI:EVENT:INP:LAN:FILT? "LAN0"

## :LXI:EVENT:INPut:LAN[:SET]:IDENtifier

Specifies the string that is expected to arrive over the LAN for a given input event to occur. For example, "Relay Closed" as an identifier for "LAN0" sent from a switch box.

### Syntax

:LXI:EVENT:INPut:LAN[:SET]:IDENtifier *strLanEvent*, *strCustomId*

:LXI:EVENT:INPut:LAN[:SET]:IDENtifier? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***strCustomId*** Identifier. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

### Query response

*strCustomId* <newline>

*strCustomId* returns a string with the identifier assigned to the specified input event. Response data type is SRD.

**Example**                   :LXI:EVEN:INP:LAN:IDEN "LAN0","MyEvent"  
                          :LXI:EVEN:INP:LAN:IDEN? "LAN0"

## **:LXI:EVENT:LOG:ALL?**

Returns the contents of the event log. A complete log is a set of entries delimited by a semicolon. The contents of the event log are returned as a single quoted string containing all the event log entries. This is a non-destructive read.

**Syntax**                   :LXI:EVENT:LOG:ALL?

**Query response**       *log* <newline>  
*log* returns the contents of the event log. Response data type is SRD.

**Example**                   :LXI:EVEN:LOG:ALL?

## **:LXI:EVENT:LOG:CIRCular[:ENABLE]**

Selects how new entries are handled when the LXI event log is full.

**Syntax**                   :LXI:EVENT:LOG:CIRCular[:ENABLE] *status*  
                          :LXI:EVENT:LOG:CIRCular[:ENABLE]?

**Parameter**           *status*                   Circular logging status. 0|OFF|1|ON (default). Parameter data type is boolean.

**Query response**       *status* <newline>  
*status* returns an integer (0 or 1) indicating the behavior of the LXI event log. Response data type is boolean.

**Example**                   :LXI:EVEN:LOG:CIRC ON  
                          :LXI:EVEN:LOG:CIRC?

## **:LXI:EVENT:LOG:CIRCular:FBEntry**

Selects the most recently added event log entry to be used as the reference for **:LXI:EVENT:LOG:ENTRy?**. The log must be in circular mode for the command to function.

**Syntax** :LXI:EVENT:LOG:CIRCular:FBEntry

**Example** :LXI:EVEN:LOG:CIRC:FBE

## **:LXI:EVENT:LOG:CLEar**

Removes all existing entries from the event log.

**Syntax** :LXI:EVENT:LOG:CLEar

**Example** :LXI:EVEN:LOG:CLE

## **:LXI:EVENT:LOG:COUNT?**

Returns an integer as the total number of entries in the LXI event log.

**Syntax** :LXI:EVENT:LOG:COUNT?

**Query response** *count* <newline>

*count* returns an integer as the total number of entries in the LXI event log. Response data type is NR1.

**Example** :LXI:EVEN:LOG:COUN?

## **:LXI:EVENT:LOG:ENABle**

Enables or disables LXI event logging.

**Syntax** :LXI:EVENT:LOG:ENABle *status*

:LXI:EVENT:LOG:ENABle?

**Parameter** *status* LXI event logging status. 0|OFF|1|ON (default). Parameter data type is boolean.

**Query response** *status* <newline>

*status* returns an integer (0 or 1) indicating the behavior of the LXI event log.  
Response data type is boolean.

**Example**

```
:LXI:EVEN:LOG:ENAB ON  
:LXI:EVEN:LOG:ENAB?
```

## **:LXI:EVENT:LOG:ENTRy?**

Retrieves the event log entry referenced by <intIndex>. When the log is in circular mode, this index value is relative to the entry selected by  
**:LXI:EVENT:LOG:CIRCular:FBEntry**.

**Syntax**

```
:LXI:EVENT:LOG:ENTRy? intIndex
```

**Parameter**

|                 |  |
|-----------------|--|
| <i>intIndex</i> | Reference point of the event log. 0 to 2147483647. Parameter data type is NRf. |
|-----------------|--|

**Query response**

```
log <newline>
```

*log* returns an event log entry, nine comma separated character strings, as follows.  
“<chrDate>,<chrTime>,<chrEventType>,<chrEventName>,<chrEventEdge>,<chrSourceEvent>,<chrEventIdentifier>,<chrSrcAddress>,<chrDstAddress>”

**Example**

```
:LXI:EVEN:LOG:ENTR? 100
```

This example returns the 100th entry in the event log.

**Remarks**

The command returns an empty string if the specified index is out of range or the entry no longer exists.

The event log records internal status events as well as all LXI event activity. As LXI LAN events are sent or received, the activity is noted in the event log with an IEEE 1588 timestamp.

The fields recorded in the event log are:

- The date the event occurred (GMT).
- The time the event occurred (GMT).
- The type of event: LAN Input, LAN Output, Status, Alarm, Trigger Alarm, Trigger LAN.
- The name of the event.



- The edge associated with the event.
- The source event is only valid for LAN Output, Trigger LAN, and Trigger Alarm event types.
- The event's identifier appears as an ASCII character on the LAN.
- The source address is only valid for LAN Input event types. It is the address from which the message originated.
- The destination address is only valid for LAN Output event types. It is the address (or addresses) that the message will be sent to. For UDP messages, this field will read "ALL".

See also **“:LXI:EVENT:LOG:CLEAr”** on page 4-49.

## **:LXI:EVENT:LOG:SIZE**

Sets the maximum number of entries the LXI event log can hold.

### **Syntax**

:LXI:EVENT:LOG:SIZE *maxLogEntries*

:LXI:EVENT:LOG:SIZE?

### **Parameter**

***maxLogEntries***      Size of the LXI event log. 0 to 200. Parameter data type is NRf.

### **Query response**

*maxLogEntries* <newline>

*maxLogEntries* returns an integer as the current size (maximum number of entries) of the LXI event log. Response data type is NR1.

### **Example**

:LXI:EVENT:LOG:SIZE 200

:LXI:EVENT:LOG:SIZE?

## **:LXI:EVENT[:OUTPut]:LAN:ADD**

B2900 ignores this command. Creates a new output event as specified by <strLanEvent>.

### **Syntax**

:LXI:EVENT[:OUTPut]:LAN:ADD *strLanEvent*

### **Parameter**

***strLanEvent***      LAN event name. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

**Example** :LXI:EVEN:LAN:ADD “MyEvent”

### **:LXI:EVENT[:OUTPut]:LAN:COUNt?**

Returns an integer as the number of configured LXI output LAN events.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:COUNt?

**Query response** *count* <newline>

*count* returns an integer as the total number of defined output LAN events. Response data type is NR1.

**Example** :LXI:EVEN:LAN:COUN?

### **:LXI:EVENT[:OUTPut]:LAN:DISable:ALL**

Disables all configured LXI output LAN events.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:DISable:ALL

**Example** :LXI:EVEN:LAN:DIS:ALL

### **:LXI:EVENT[:OUTPut]:LAN:LIST?**

Returns a quoted string containing a list of all configured LAN output event names.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:LIST?

**Query response** *list* <newline>

*list* returns a quoted string with the list of defined output event names. Response data type is SRD.

**Example** :LXI:EVEN:LAN:LIST?

### **:LXI:EVENT[:OUTPut]:LAN:REMOve:ALL**

B2900 ignores this command. Disables and removes all custom LAN events added using **:LXI:EVENT[:OUTPut]:LAN:ADD**. Predefined LAN names (LAN0 - LAN7) cannot be removed.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:REMOve:ALL

**Example** :LXI:EVEN:LAN:REM:ALL

## **:LXI:EVENT[:OUTPut]:LAN:REMove**

B2900 ignores this command. Disables and removes the specified custom LAN output event. Predefined LAN event names (LAN0 - LAN7) cannot be removed.

The event is added by :LXI:EVENT[:OUTPut]:LAN:ADD.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:REMove *strLanEvent*

**Parameter** *strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

**Example** :LXI:EVEN:LAN:REM “MyEvent”

## **:LXI:EVENT[:OUTPut]:LAN:SEND**

Forces the instrument to send the specified output event. The output event must be enabled, otherwise this command is ignored; in other words, you must send :LXI:EVENT[:OUTPut]:LAN[:SET]:ENABLE <strLanEvent>, ON to enable the lanEvent prior to sending SEND, or SEND will not send anything.

**Syntax** :LXI:EVENT[:OUTPut]:LAN:SEND *strLanEvent,type*

**Parameter** *strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

*type* Event type. RISE|FALL. Parameter data type is CPD.

**Example** :LXI:EVEN:LAN:SEND “LAN0”,FALL

## **:LXI:EVENT[:OUTPut]:LAN[:SET]:CONFigure**

Configures the most common attributes of LXI LAN output events. The parameters are also available from their corresponding commands. For more details on the configurable parameters, refer to the individual SCPI commands below.

**Syntax** :LXI:EVENT[:OUTPut]:LAN[:SET]:CONFigure *strLanEvent, enable, source, slope, drive, destination*

**Parameter** *strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

|                           |   |
|---------------------------|---|
| <i><b>enable</b></i>      | Status. 0 OFF 1 ON. Parameter data type is boolean.   |
| <i><b>source</b></i>      | Event name, should be one of following. Parameter data type is SPD. <ul style="list-style-type: none"><li>• WaitingForAcquireArm1</li><li>• WaitingForAcquireArm2</li><li>• WaitingForAcquireTrigger1</li><li>• WaitingForAcquireTrigger2</li><li>• WaitingForTransitionArm1</li><li>• WaitingForTransitionArm2</li><li>• WaitingForTransitionTrigger1</li><li>• WaitingForTransitionTrigger2</li><li>• Measuring1</li><li>• Measuring2</li><li>• Settling1</li><li>• Settling2</li></ul> |
| <i><b>slope</b></i>       | Slope attribute. POSitive NEGative. Parameter data type is CPD.   |
| <i><b>drive</b></i>       | Drive behavior. OFF NORMal WOR. Parameter data type is CPD. See “:LXI:EVENT[:OUTPut]:LAN[:SET]:DRIVE” on <a href="#">page 4-55</a> .  |
| <i><b>destination</b></i> | Destination. String enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "*", etc. Parameter data type is SPD.  |

#### Example

:LXI:EVEN:LAN:CONF “LAN0”,ON,”WaitingForAcquireTrigger1”,POS,NORM,  
”ALL:5044”

### **:LXI:EVENT[:OUTPut]:LAN[:SET]:DESTination**

Sets the destination for the specified outgoing LAN event to the hosts specified by <strDestExpression>. The expression takes the form of “host1:port1, host2:port2, ”. The port numbers are optional and will default to the IANA assigned TCP port (5044).

**Syntax** :LXI:EVENT[:OUTPut]:LAN[:SET]:DESTination *strLanEvent*, *strDestExpression*  
:LXI:EVENT[:OUTPut]:LAN[:SET]:DESTination? *strLanEvent*

**Parameter**      *strLanEvent*      LAN event name. String enclosed in quotes. Parameter data type is SPD.  
  
*strDestExpression*      Destination. String enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

**Query response**      *strDestExpression* <newline>  
  
*strDestExpression* returns the Destination Expression string. Response data type is SRD.

**Example** :LXI:EVEN:LAN:DEST “LAN0”,”ALL:5044”  
:LXI:EVEN:LAN:DEST? “LAN0”

## :LXI:EVENT[:OUTPut]:LAN[:SET]:DRIVE

Specifies the trigger drive behavior for the specified LAN output event.

**Syntax** :LXI:EVENT[:OUTPut]:LAN[:SET]:DRIVE *strLanEvent*, *drive*  
:LXI:EVENT[:OUTPut]:LAN[:SET]:DRIVE? *strLanEvent*

**Parameter**      *strLanEvent*      LAN event name. String enclosed in quotes. Parameter data type is SPD.  
  
*drive*      Drive behavior. OFF (default)|NORMal|WOR. Parameter data type is CPD.  
  
OFF disables the LAN event.  
  
NORMal designates typical operation where both edges of the instrument event are transmitted.  
  
WOR (Wired.OR) causes only one edge to be transmitted.

**Query response**      *drive* <newline>  
  
*drive* returns a string representing the drive behavior. Response data type is CRD.

**Example** :LXI:EVEN:LAN:DRIV “LAN0”,WOR  
:LXI:EVEN:LAN:DRIV? “LAN0”

## :LXI:EVENT[:OUTPut]:LAN[:SET]:ENABle

Enables or disables the specified LXI LAN output event.

### Syntax

:LXI:EVENT[:OUTPut]:LAN[:SET]:ENABle *strLanEvent*, *status*

:LXI:EVENT[:OUTPut]:LAN[:SET]:ENABle? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***status*** LXI LAN output event status. 0|OFF (default)|1|ON. Parameter data type is boolean.

### Query response

*status* <newline>

*status* returns an integer (0 or 1) as the state of the specified output event. Response data type is boolean.

### Example

:LXI:EVEN:LAN:ENAB "LAN0",ON

:LXI:EVEN:LAN:ENAB? "LAN0"

## :LXI:EVENT[:OUTPut]:LAN[:SET]:IDENtifier

Specifies the custom string that will be transmitted as part of the output event.

### Syntax

:LXI:EVENT[:OUTPut]:LAN[:SET]:IDENtifier *strLanEvent*, *strCustomId*

:LXI:EVENT[:OUTPut]:LAN[:SET]:IDENtifier? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***strCustomId*** Identifier. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

### Query response

*strCustomId* <newline>

*strCustomId* returns a string with the identifier assigned to the specified LXI LAN output event. Response data type is SRD.

### Example

:LXI:EVEN:LAN:IDEN "LAN0","MyEvent"

:LXI:EVEN:LAN:IDEN? "LAN0"

## :LXI:EVENT[:OUTPut]:LAN[:SET]:SLOPe

Sets the slope of the event transition. It determines which instrument event transition will result in a LAN packet being sent and whether or not that edge is inverted.

### Syntax

:LXI:EVENT[:OUTPut]:LAN[:SET]:SLOPe *strLanEvent*, *slope*

:LXI:EVENT[:OUTPut]:LAN[:SET]:SLOPe? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***slope*** Slope attribute. POSitive (default)|NEGative. Parameter data type is CPD.

### Query response

*slope* <newline>

*slope* returns a character string representing the slope attribute of the LXI LAN packet. Response data type is CRD.

### Example

:LXI:EVEN:LAN:SLOP "LAN0",POS

:LXI:EVEN:LAN:SLOP? "LAN0"

## :LXI:EVENT[:OUTPut]:LAN[:SET]:SOURce

Designates the instrument that the specified LAN output event is tied to. <strAnyEvent> is a discrete value.

### Syntax

:LXI:EVENT[:OUTPut]:LAN[:SET]:SOURce *strLanEvent*, *strAnyEvent*

:LXI:EVENT[:OUTPut]:LAN[:SET]:SOURce? *strLanEvent*

### Parameter

***strLanEvent*** LAN event name. String enclosed in quotes. Parameter data type is SPD.

***strAnyEvent*** Event name, should be one of following. Parameter data type is SPD.

- WaitingForAcquireArm1
- WaitingForAcquireArm2
- WaitingForAcquireTrigger1
- WaitingForAcquireTrigger2
- WaitingForTransitionArm1

- WaitingForTransitionArm2
- WaitingForTransitionTrigger1
- WaitingForTransitionTrigger2
- Measuring1
- Measuring2
- Settling1
- Settling2

**Query response**     *strAnyEvent* <newline>  
*strAnyEvent* returns a character string indicating the event type for the specified LAN event name. Response data type is SRD.

**Example**     :LXI:EVEN:LAN:SOUR “LAN0”,”Measuring1”  
                 :LXI:EVEN:LAN:SOUR? “LAN0”

## **:LXI:EVENT[:OUTPut]:LAN[:SET]:TSDelta**

Sets the delay that occurs between the generation of the specified event and the remote instruments action on it. The parameter <dblSeconds> represents a time in seconds to add to the timestamp of the output LAN event. Since precision timed interrupts (also known as alarms) are not supported at this time, <dblSeconds> must be 0.0.

**Syntax**     :LXI:EVENT[:OUTPut]:LAN[:SET]:TSDelta *strLanEvent*, *dblSeconds*  
                 :LXI:EVENT[:OUTPut]:LAN[:SET]:TSDelta? *strLanEvent*

**Parameter**     *strLanEvent*     LAN event name. String enclosed in quotes. Parameter data type is SPD.  
*dblSeconds*     Delay, in seconds. Parameter data type is NRf.

**Query response**     *dblSeconds* <newline>  
*dblSeconds* returns the delay, in seconds. Response data type is NR3.

**Example**     :LXI:EVEN:LAN:TSD “LAN0”,1.0  
                 :LXI:EVEN:LAN:TSD? “LAN0”



## **: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:HNAME?

## **: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:SNAME:DES "B2900"

:LXI:MDNS:SNAME: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:SNAME?

## :TRIGger:LXI:LAN[:SET]:DELay

Optional command. <delay> is a double representing the delay, in seconds. Any <delay> value other than 0.0 is changed to 0 (zero) and a warning is inserted into the SCPI error queue.

### NOTE

This implementation assumes that there is no Precision Time Protocol (PTP) implementation on the instrument. Delay requires PTP, so without it the delay must be zero.

### Syntax

```
:TRIGger:LXI:LAN[:SET]:DELay delay
:TRIGger:LXI:LAN[:SET]:DELay?
```

### Parameter

*delay* Delay, in seconds. Parameter data type is NRf.

### Query response

*delay* <newline>  
*delay* returns the delay, in seconds. Response data type is NR3.

### Example

```
:TRIG:LXI:LAN:DEL 10.0
:TRIG:LXI:LAN:DEL?
```

## :TRIGger:LXI:LAN[:SET]:DETection

Specifies the behavior of the trigger signal.

### Syntax

```
:TRIGger:LXI:LAN[:SET]:DETection strLanEvent, detection
:TRIGger:LXI:LAN[:SET]:DETection? strLanEvent
```

### Parameter

*strLanEvent* LAN event name. String enclosed in quotes. Parameter data type is SPD.

*detection* Detection type. RISE (default)|FALL|HIGH|LOW. Parameter data type is CPD.

RISE sets the instrument to trigger on the receipt of a signal LOW LAN event followed by a signal HIGH LAN event (rising edge).

FALL sets the instrument to trigger on the receipt of a signal HIGH LAN event followed by a signal LOW LAN event (falling edge).

HIGH sets the instrument to trigger on every signal HIGH LAN event.

LOW sets the instrument to trigger on every signal LOW LAN event.

**Query response**     *detection* <newline>  
*detection* returns a string with the DETection setting for the specified event.  
Response data type is CRD.

**Example**     :TRIG:LXI:LAN:DET "LAN0",RISE  
              :TRIG:LXI:LAN:DET? "LAN0"

## **:TRIGger:LXI:LAN[:SET]:ENABLE**

Enables or disables the specified LAN trigger. When enabled the instrument will trigger upon receiving any event from the LXI trigger LAN event list.

**Syntax**     :TRIGger:LXI:LAN[:SET]:ENABle *strLanEvent*, *status*  
              :TRIGger:LXI:LAN[:SET]:ENABle? *strLanEvent*

**Parameter**     *strLanEvent*     LAN event name. String enclosed in quotes. Parameter data type is SPD.  
*status*     Status. 0|OFF (default)|1|ON. Parameter data type is boolean.

**Query response**     *status* <newline>  
*status* returns an integer (0 or 1) denoting the enabled status of the specified event.  
Response data type is boolean.

**Example**     :TRIG:LXI:LAN:ENAB "LAN0",1  
              :TRIG:LXI:LAN:ENAB? "LAN0"

## **:TRIGger:LXI:LAN[:SET]:FILTer**

Allows user to create a filter expression for the specified LAN trigger event. Only LXI trigger LAN events coming from hosts matching the filter string are processed.

**Syntax**     :TRIGger:LXI:LAN[:SET]:FILTer *strLanEvent*, *strFilterExpression*  
              :TRIGger:LXI:LAN[:SET]:FILTer? *strLanEvent*

**Parameter**     *strLanEvent*     LAN event name. String enclosed in quotes. Parameter data type is SPD.

### ***strFilterExpression***

Filter. String enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "\*", etc. Parameter data type is SPD.

**Query response**     *strFilterExpression* <newline>

*strFilterExpression* returns a string with the filter expression for the specified event. Response data type is SRD.

**Example**     :TRIG:LXI:LAN:FILT "LAN0","ALL:5024"  
:TRIG:LXI:LAN:FILT? "LAN0"

## **:TRIGger:LXI:LAN[:SET]:IDENtifier**

Sets the string that is expected to arrive over the LAN for a given trigger LAN event to occur.

**Syntax**     :TRIGger:LXI:LAN[:SET]:IDENtifier *strLanEvent*, *strCustomId*  
:TRIGger:LXI:LAN[:SET]:IDENtifier? *strLanEvent*

|                  |                           |  |
|------------------|---------------------------|--|
| <b>Parameter</b> | <b><i>strLanEvent</i></b> | LAN event name. String enclosed in quotes. Parameter data type is SPD.   |
|                  | <b><i>strCustomId</i></b> | Identifier. String of up to 16 characters enclosed in quotes. You can use letters (A–Z, a–z), numbers (0–9), and printable characters like "@", "%", "*", etc. Parameter data type is SPD. |

**Query response**     *strCustomId* <newline>

*strCustomId* returns the identifier string for the specified event. Response data type is SRD.

**Example**     :TRIG:LXI:LAN:IDEN "LAN0","MyEvent"  
:TRIG:LXI:LAN:IDEN? "LAN0"

## MEASure Subsystem

### :MEASure?

Executes a spot (one-shot) measurement for the parameters specified by the :SENSe:FUNCTION[:ON] command, and returns the measurement result data specified by the :FORMat:ELEMents:SENSe command. Measurement conditions must be set by SCPI commands or front panel operation before executing this 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

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

## :MEASure:<CURRENT|RESistance|VOLTage>?

Executes a spot (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

```
:SENS:FUNC CURR
: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 extension 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 extension 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 extension 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**

|                  |  |
|------------------|--|
| <i>macro</i>     | Name of macro.   |
| <i>file_name</i> | Name of the file used to save the macro. File extension must be <i>mac</i> , 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**

|                  |   |
|------------------|---|
| <i>file_name</i> | Name of the file used to save the instrument setup. Up to 255 ASCII characters. Parameter data type is SPD. File extension must be <i>sta</i> , 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**

|                  |   |
|------------------|---|
| <i>file_name</i> | Name of the file used to save the specified data. Up to 255 ASCII characters. Parameter data type is SPD. File extension must be <i>tra</i> , which is meaningful for the :MMEMory:CATalog? result. |
|------------------|---|

## Subsystem Commands

### MMEMory Subsystem

***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  $\mu$ s 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.

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <i>mode</i> <newline><br><br><i>mode</i> is 0 or 1, and indicates that the high capacitance mode is off or on, respectively. Response data type is NR1.  |
| <b>Example</b>        | :OUTP:HCAP 1<br>:OUTP2:HCAP:STAT?<br><br><b>:OUTPut:LOW</b><br><br>Selects the state of the low terminal. Before executing this command, the source output must be disabled by the <b>:OUTPut[:STATe]</b> command. Or else, an error occurs. |
| <b>Syntax</b>         | :OUTPut[ <i>c</i> ]:LOW <i>low_state</i><br>:OUTPut[ <i>c</i> ]:LOW?   |
| <b>Parameter</b>      | <i>low_state</i> FLOat GROund (default). Parameter data type is CPD.<br><i>low_state</i> =FLOat sets the floating state.<br><i>low_state</i> =GROund sets the ground state. The low terminal is connected to ground.                         |
| <b>Query response</b> | <i>low_state</i> <newline><br><br><i>low_state</i> is FLO or GRO, and indicates the low terminal state. Response data type is CRD.   |
| <b>Example</b>        | :OUTP:LOW FLO<br>:OUTP2:LOW?<br><br><b>:OUTPut:OFF:AUTO</b><br><br>Enables or disables the automatic output off function.  |
| <b>Syntax</b>         | :OUTPut[ <i>c</i> ]:OFF:AUTO <i>mode</i><br>:OUTPut[ <i>c</i> ]:OFF:AUTO?  |
| <b>Parameter</b>      | <i>mode</i> 1 ON 0 OFF (default). Parameter data type is boolean.<br><i>mode</i> =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  $\mu$ A at the 100  $\mu$ A 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  $\mu$ A at the 100  $\mu$ A 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.

|                       |   |
|-----------------------|---|
| <b>Syntax</b>         | :OUTPut[ <i>c</i> ]:PROTection[:STATe] <i>mode</i><br>:OUTPut[ <i>c</i> ]:PROTection[:STATe]?   |
| <b>Parameter</b>      | <i>mode</i> 1 ON 0 OFF (default). Parameter data type is boolean.<br><i>mode</i> =0 or OFF disables the over voltage/current protection.<br><i>mode</i> =1 or ON enables the over voltage/current protection. |
| <b>Query response</b> | <i>mode</i> <newline><br><i>mode</i> is 0 or 1, and indicates that the over voltage/current protection is off or on, respectively. Response data type is NR1.   |
| <b>Example</b>        | :OUTP:PROT 1<br>:OUTP2:PROT:STAT?   |

## :OUTPut:RECall

Recalls the channel setup saved by the :OUTPut:SAVE command.

|                  |   |
|------------------|---|
| <b>Syntax</b>    | :OUTPut[ <i>c</i> ]:RECall <i>index</i>   |
| <b>Parameter</b> | <i>index</i> 0 1. Parameter data type is NR1.<br><i>index</i> =0 is used to recall channel setup 0.<br><i>index</i> =1 is used to recall channel setup 1. |
| <b>Example</b>   | :OUTP:REC 1   |

## :OUTPut:SAVE

Saves the channel setup. The setup can be recalled by the :OUTPut:RECall command.

|               |                                       |
|---------------|---------------------------------------|
| <b>Syntax</b> | :OUTPut[ <i>c</i> ]:SAVE <i>index</i> |
|---------------|---------------------------------------|

Subsystem Commands  
OUTPut Subsystem

|                       |  |
|-----------------------|--|
| <b>Parameter</b>      | <p><i>index</i> 0 1. Parameter data type is NR1.</p> <p><i>index</i>=0 is used to memorize the present channel setup as channel setup 0.</p> <p><i>index</i>=1 is used to memorize the present channel setup as channel setup 1.</p> |
| <b>Example</b>        | <p>:OUTP:SAVE 1</p> <p><b>:OUTPut[:STATe]</b></p> <p>Enables or disables the source output.</p>  |
| <b>Syntax</b>         | <p>:OUTPut[<i>c</i>][:STATe] <i>mode</i></p> <p>:OUTPut[<i>c</i>][:STATe]?</p>   |
| <b>Parameter</b>      | <p><i>mode</i> 1 ON 0 OFF (default). Parameter data type is boolean.</p> <p><i>mode</i>=1 or ON enables the source output.</p> <p><i>mode</i>=0 or OFF disables the source output.</p>   |
| <b>Query response</b> | <p><i>mode</i> &lt;newline&gt;</p> <p><i>mode</i> is 0 or 1, and indicates that the source output is off or on, respectively.</p> <p>Response data type is NR1.</p>  |
| <b>Example</b>        | <p>:OUTP 1</p> <p>:OUTP2:STAT?</p>   |

## 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 **:PROG**ram[: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 **:PROG:VARiable** command before executing the program.

**Example**

```
:PROG:NAME "program1"  
:PROG:DEF #220:SOUR:FUNC:MODE CURR
```

## **:PROG:SELected]:DELe: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**

```
:PROG:SELected]:DELe:ALL
```

**Example**

```
:PROG:DEL:ALL
```

## **:PROG:SELected]:DELe:SELected]**

Deletes a program stored in the program memory.

Before executing this command, the program must be selected by the **:PROG: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**

```
:PROG:SELected]:DELe:SELected]
```

**Example**

```
:PROG:NAME "program1"  
:PROG:DEL
```

## **:PROG:SELected]:EXECute**

Executes a program stored in the program memory.

Before executing this command, the program must be selected by the **:PROG: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**

```
:PROG: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

| <i>operation</i> | 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,+9.910  
000E+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.

## Subsystem Commands

### READ Subsystem

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*.

$$nplc = time \times power\ line\ frequency$$

**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.

Subsystem Commands  
SENSe Subsystem

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][:BOTH]**

Sets the compliance value of the specified channel. The setting value is applied to both positive and negative sides.

Keyword BOTH is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.

**Syntax**             :SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel][:BOTH]  
                      *compliance*  
  
                      :SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel][:BOTH]?  
                      [*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  $\mu$ A 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[:LEVel]:NEGative**

This command is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.

Sets the negative side compliance value of the specified channel.

### **Syntax**

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]:NEGative  
*compliance*

:SENSe[*c*]:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]:NEGative?  
[*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  $\mu$ A 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:NEG -1E-3

:SENS2:CURR:DC:PROT:NEG?

## **:SENSe:<CURRent[:DC]|VOLTage[:DC]>:PROTection[:LEVel]:POSitive**

This command is for the B2900 with the license “SWS” and the firmware revision 3.1 or later.

Sets the positive side compliance value of the specified channel.

|                       |  |  |
|-----------------------|--|--|
| <b>Syntax</b>         | <pre>:SENSe[<i>c</i>]:&lt;CURREnt[:DC] VOLTage[:DC]&gt;:PROTection[:LEVel]:POSitive <i>compliance</i></pre> <pre>:SENSe[<i>c</i>]:&lt;CURREnt[:DC] VOLTage[:DC]&gt;:PROTection[:LEVel]:POSitive? [<i>compliance</i>]</pre> <p>For &lt;CURREnt[:DC] VOLTage[:DC]&gt;, specify CURREnt[:DC] for current compliance or VOLTage[:DC] for voltage compliance.</p> |  |
| <b>Parameter</b>      | <b><i>compliance</i></b>   | <i>value</i> (see maximum current or maximum voltage in “ <b>Source Output Ranges</b> ” on page 2-24)  MINimum MAXimum DEFault (default is +100 $\mu$ A or +2 V). Parameter data type is NRf+. Effective values of <i>value</i> are from the minimum measurement value to the maximum measurement value of the channel. Query does not support <i>compliance=value</i> . |
| <b>Query response</b> | <pre><i>compliance</i> &lt;newline&gt;</pre> <p><i>compliance</i> returns the present compliance value. If a parameter is specified, <i>compliance</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.</p>  |  |
| <b>Example</b>        | <pre>:SENS:CURREnt:PROTection:POSitive 1E-3</pre> <pre>:SENS2:CURREnt:DC:PROTection:POSitive?</pre><br><pre><b>:SENSe:&lt;CURREnt[:DC] VOLTage[:DC]&gt;:PROTection :TRIPped?</b></pre> <p>Returns the compliance status of the specified channel.</p>  |  |
| <b>Syntax</b>         | <pre>:SENSe[<i>c</i>]:&lt;CURREnt[:DC] VOLTage[:DC]&gt;:PROTection:TRIPped?</pre> <p>For &lt;CURREnt[:DC] VOLTage[:DC]&gt;, specify CURREnt[:DC] for current compliance, or VOLTage[:DC] for voltage compliance.</p>   |  |
| <b>Query response</b> | <pre><i>status</i> &lt;newline&gt;</pre> <p><i>status</i> is 1 or 0, and indicates if the channel is in the compliance state or not. Response data type is NR1.</p>  |  |
| <b>Example</b>        | <pre>:SENS:CURREnt:PROTection:TRIPped?</pre>   |  |



## **: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 ranging measurement. 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.

|                       |   |
|-----------------------|---|
| <b>Parameter</b>      | <p><b><i>mode</i></b> NORMal (default) RESolution SPEed. Parameter data type is CPD.</p> <p><i>mode</i>=NORMal supports basic operation and downward changing operation described below</p> <p><i>mode</i>=RESolution supports basic operation and upward changing operation described below</p> <p><i>mode</i>=SPEed supports basic operation and upward and downward changing operations described below</p> <ul style="list-style-type: none"> <li>• Basic operation<br/>The channel automatically sets the range which provides the best resolution to perform the measurement.</li> <li>• Upward changing operation<br/>If measured data <math>\geq \text{value1}</math>, the range shifts upward after a measurement.<br/><math>\text{value1} = \text{measurement range} \times \text{rate} / 100</math></li> <li>• Downward changing operation<br/>If measured data <math>\leq \text{value2}</math>, the range shifts downward immediately.<br/><math>\text{value2} = \text{measurement range} \times \text{rate} / 1000</math></li> </ul> <p><i>rate</i> value is set by the<br/>:SENSe:&lt;CURRent[:DC] VOLTage[:DC]&gt;:RANGe:AUTO:THReshold command.</p> |
| <b>Query response</b> | <p><i>mode</i> &lt;newline&gt;</p> <p><i>mode</i> returns NORM, RES, or SPE, and indicates that the operation mode of automatic measurement ranging. Response data type is CRD.</p>   |
| <b>Example</b>        | <pre>:SENS:CURR:RANG:AUTO:MODE SPE :SENS2:CURR:DC:RANG:AUTO:MODE?  <b>:SENSe:&lt;CURRent[:DC] VOLTage[:DC]&gt;:RANGe:AUTO:THReshold</b></pre> <p>Sets the threshold rate for the automatic measurement ranging operation.</p>   |
| <b>Syntax</b>         | <pre>:SENSe[c]:&lt;CURRent[:DC] VOLTage[:DC]&gt;:RANGe:AUTO:THReshold <i>rate</i> :SENSe[c]:&lt;CURRent[:DC] VOLTage[:DC]&gt;:RANGe:AUTO:THReshold? [<i>rate</i>]</pre>   |

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*+1th 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 4-108](#) 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 4-108 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:

$\text{wait time} = \text{gain} \times \text{initial wait time} + \text{offset}$

- **:SENSe:WAIT:AUTO OFF|0** condition:

$\text{wait time} = \text{offset}$

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*], [*i*], 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 from the trigger source set by the `:TRIGger<:ACQuire[:TRANSient[:ALL]]>:SOURce[:SIGNal]` command.

For the output level after the trigger status is returned to idle, see “`[[:SOURce]:FUNCTION:TRIGgered:CONTinuous]`” on page 4-127.

**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:CURRE: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:CURRE: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* (see “[Sweep Steps](#)” on page 2-28) |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.

$stop = start + step \times (points - 1)$

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 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.

|   |  |  |
|---|--|--|
| <b>Syntax</b>   | [:SOURce[c]]:<CURRent VOLTAge>:RANGe:AUTO:LLIMit <i>range</i><br>[:SOURce[c]]:<CURRent VOLTAge>:RANGe:AUTO:LLIMit? [ <i>range</i> ]<br>For <CURRent VOLTAge>, specify CURRent for current output, or VOLTAge for voltage output. |  |
| <b>Parameter</b>  | <i>range</i>   | <i>value</i> (see “Source Output Ranges” on page 2-24)  MINimum MAXimum DEFault. Parameter data type is NRf+. Query does not support <i>range=value</i> .<br><br><i>value</i> for current output: See Table 2-6.<br><i>value</i> for voltage output: See Table 2-4.  |
| <b>Query response</b>   | <i>range</i> <newline><br><br><i>range</i> returns the present setting. If a parameter is specified, <i>range</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.                                     |  |
| <b>Example</b>  | :CURR:RANG:AUTO:LLIM 1E-6<br>:SOUR2:VOLT:RANG:AUTO:LLIM?   |  |
| <b>[[:SOURce]:&lt;CURRent VOLTAge&gt;:RANGe:RPRiority</b><br>Selects the output pulse priority mode, low noise priority or transient speed priority, of the specified source channel. |  |  |
| <b>Syntax</b>   | [:SOURce[c]]:<CURRent VOLTAge>:RANGe:RPRiority <i>mode</i><br>[:SOURce[c]]:<CURRent VOLTAge>:RANGe:RPRiority?<br>For <CURRent VOLTAge>, specify CURRent for current output, or VOLTAge for voltage output.                       |  |
| <b>Parameter</b>  | <i>mode</i>  | NOISe (default) TRANsient. Parameter data type is CPD.<br><br><i>mode</i> =NOISe sets the low noise priority mode to obtain clean output pulse.<br><br><i>mode</i> =TRANsient sets the transient speed priority mode to obtain high slew rate of output pulse. This mode is effective when the 1 A, 1.5 A or 10 A range is used for the current output or the current measurement. |
| <b>Query response</b>   | <i>mode</i> <newline><br><br><i>mode</i> returns NOIS or TRAN. Response data type is CRD.  |  |

**Example** :CURR:RANG:RPR TRAN  
:SOUR2:VOLT:RANG:RPR?

## **[[: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.

$stop = start + step \times (points - 1)$

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]]:<CURRENT|VOLTage>:TRANSient:SPEed**

Selects the transient speed mode, normal or fast, of the specified source channel.

## Syntax

[[:SOURce[c]]:<CURRENT|VOLTage>:TRANSient:SPEed *mode*

[[:SOURce[c]]:<CURRENT|VOLTage>:TRANSient:SPEed?

## Subsystem Commands

### SOURce Subsystem

For <CURREnt|VOLTage>, specify CURREnt for current output, or VOLTage for voltage output.

**Parameter**            *mode*                      NORMal (default)|FAST. Parameter data type is CPD.  
*mode*=NORMal sets the normal mode to obtain clean output.  
*mode*=FAST sets the fast mode to obtain high slew rate of output. This mode is not effective when the high capacitance mode is enabled.

**Query response**    *mode* <newline>  
*mode* returns NORM or FAST. Response data type is CRD.

**Example**                :CURR:TRAN:SPE FAST  
                          :SOUR2:VOLT:TRAN:SPE?

## **[[: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:FUNCTION**

Assigns the input/output function to the specified GPIO pin.

**Syntax**                [[:SOURce]:DIGital:EXternal[*n*][:FUNCTION] *function*  
                          [[:SOURce]:DIGital:EXternal[*n*][:FUNCTION]?



|                  |                        |   |
|------------------|------------------------|---|
| <b>Parameter</b> | <b><i>function</i></b> | Function. DINPut (default for the EXT1 to EXT13 pins) DIO HVOL (default for the EXT14 pin) TINPut TOUT. Parameter data type is CPD. |
|                  | <i>function</i> =DINP  | assigns the digital input.  |
|                  | <i>function</i> =DIO   | assigns the digital I/O.  |
|                  | <i>function</i> =HVOL  | assigns the high voltage status output. Only for the EXT14 pin.   |
|                  | <i>function</i> =TINP  | assigns the trigger input.  |
|                  | <i>function</i> =TOUT  | assigns the trigger output.   |

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <i>function</i> <newline>  |
|                       | <i>function</i> returns DIO, DINP, TOUT, or TINP. Response data type is CRD. |

|                |  |
|----------------|--|
| <b>Example</b> | :DIG:EXT TOUT<br>:SOUR:DIG:EXT14:FUNC? |
|----------------|--|

## **[[:SOURce]:DIGital:EXTernal: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:FUNCTION** command

|               |   |
|---------------|---|
| <b>Syntax</b> | [[:SOURce]:DIGital:EXTernal[ <i>n</i> ]:POLarity <i>polarity</i><br>[[:SOURce]:DIGital:EXTernal[ <i>n</i> ]:POLarity? |
|---------------|---|

|                  |                        |  |
|------------------|------------------------|--|
| <b>Parameter</b> | <b><i>polarity</i></b> | 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. |
|                  | <i>polarity</i> =POS   | sets positive polarity.  |
|                  | <i>polarity</i> =NEG   | sets negative polarity.  |

|                       |  |
|-----------------------|--|
| <b>Query response</b> | <i>polarity</i> <newline>                                      |
|                       | <i>polarity</i> returns POS or NEG. Response data type is CRD. |

|                |  |
|----------------|--|
| <b>Example</b> | :DIG:EXT:POL NEG<br>:SOUR:DIG:EXT14:POL? |
|----------------|--|

## **[[:SOURce]:DIGital:EXTernal:TOUTput[:EDGE]:POSi tion**

Selects the trigger output 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:TOUTput[:EDGE]:WID Th**

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: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:TOUTput[:EDGE]:POSit ion**

Selects the trigger output timing for the internal port 1 or 2.

**Syntax** [:SOURce]:DIGital:INTernal[*i*]:TOUTput[:EDGE]:POSition *position*  
[:SOURce]:DIGital:INTernal[*i*]: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).

Subsystem Commands  
SOURce Subsystem

*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:INT2: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][:BOTH]” on page 4-96.

**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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | [:SOURce[ <i>c</i> ]:FUNCTion[:SHAPE] <i>shape</i><br>[:SOURce[ <i>c</i> ]:FUNCTion[:SHAPE]? |   |
| <b>Parameter</b>      | <i>shape</i>   | Source output shape. PULSe DC (default). Parameter data type is CPD.<br><br><i>shape</i> =DC sets the specified channel to DC (constant) output.<br><i>shape</i> =PULS sets the specified channel to pulsed output. |
| <b>Query response</b> | <i>shape</i> <newline><br><i>shape</i> returns DC or PULS. Response data type is CRD.        |   |
| <b>Example</b>        | :FUNC PULS<br>:SOUR2:FUNC:SHAP?  |   |

## **[:SOURce]:FUNCTion:TRIGgered:CONTInuous**

Enables or disables continuous trigger output for the specified channel.

|                       |   |  |
|-----------------------|---|--|
| <b>Syntax</b>         | [:SOURce[ <i>c</i> ]:FUNCTion:TRIGgered:CONTInuous <i>mode</i><br>[:SOURce[ <i>c</i> ]:FUNCTion:TRIGgered:CONTInuous?                             |  |
| <b>Parameter</b>      | <i>mode</i>   | 0 OFF (default) 1 ON. Parameter data type is boolean.<br><br><i>mode</i> =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.<br><br><i>mode</i> =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 VOLTag>[:LEVel][:IMMediate][:AMPLitude] command and the range setup command. |
| <b>Query response</b> | <i>mode</i> <newline><br><i>mode</i> returns 0 or 1, and indicates that continuous trigger is off or on, respectively. Response data type is NR1. |  |
| <b>Example</b>        | :FUNC:TRIG:CONT 0<br>:SOUR2:FUNC:TRIG:CONT?   |  |

## **[[:SOURce]:LIST:<CURRENT|VOLTage>**

Sets the list sweep 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 data. Default is 0. Parameter data type is NRf.

*list* can have data up to maximum number of sweep steps (see “Sweep Steps” on page 2-28). 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 list sweep 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. The list can have data up to maximum number of sweep steps (see “Sweep Steps” on page 2-28).

### **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 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 maximum number of sweep steps (see “Sweep Steps” on page 2-28). Other index value causes an error. Default is 1. *start*=1 indicates the first data in the list (top of the list). Parameter data type is NR1.

### **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).

|                       |  |  |
|-----------------------|--|--|
| <b>Syntax</b>         | [:SOURce[ <i>c</i> ]:PULSe:DELay <i>delay</i>  |  |
|                       | [:SOURce[ <i>c</i> ]:PULSe:DELay? [ <i>delay</i> ]   |  |
| <b>Parameter</b>      | <i>delay</i>   | Delay time. <i>value</i> (0.0 to 99999.9, in seconds) MINimum MAXimum DEFault (default is 0). Parameter data type is NRf+. Query does not support <i>delay=value</i> . |
| <b>Query response</b> | <i>delay</i> <newline>   |  |
|                       | <i>delay</i> returns the present setting. If a parameter is specified, <i>delay</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3. |  |
| <b>Example</b>        | :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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | [:SOURce[ <i>c</i> ]:PULSe:WIDTh <i>width</i>      |   |
|                       | [:SOURce[ <i>c</i> ]:PULSe:WIDTh? [ <i>width</i> ] |   |
| <b>Parameter</b>      | <i>width</i>                                       | Pulse width. <i>value</i> (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 <i>width=value</i> . |
|                       |  | Minimum time for the pulse base output is also 50 $\mu$ s. And the minimum pulse period is 100 $\mu$ s.   |
| <b>Query response</b> | <i>width</i> <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* (see “Sweep Steps” on page 2-28) |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:AU**T**O]**

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:AU**T**O *mode*  
[[:SOURce[*c*]]:WAIT:AU**T**O?

**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*]

|                  |             |   |
|------------------|-------------|---|
| <b>Parameter</b> | <i>gain</i> | <i>value</i> (0 to 100) MINimum MAXimum DEFault (default is 1).<br>Parameter data type is NRf. Query does not support <i>gain=value</i> . |
|------------------|-------------|---|

|                       |   |
|-----------------------|---|
| <b>Query response</b> | <pre>gain &lt;newline&gt;</pre> <p><i>gain</i> returns the present setting of the gain value. If a parameter is specified, <i>gain</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.</p> |
|-----------------------|---|

**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]`

|                  |               |  |
|------------------|---------------|--|
| <b>Parameter</b> | <i>offset</i> | <i>value</i> (0 to 1 seconds) MINimum MAXimum DEFault (default is 0). Parameter data type is NRf. Query does not support <i>offset=value</i> . |
|------------------|---------------|--|

**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 <b>:SYSTem:LOCK:OWNer?</b> command), this bit will be set. When a remote interface releases the lock (see <b>:SYSTem:LOCK:NAME?</b> 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>

Subsystem Commands  
STATus Subsystem

*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] and [j], 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**

|                  |   |
|------------------|---|
| <i>frequency</i> | Frequency, in Hz. 55 to 6640 Hz. Parameter data type is NRf.              |
| <i>time</i>      | 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**

|             |   |
|-------------|---|
| <i>mode</i> | 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.

|                       |  |  |
|-----------------------|--|--|
| <b>Syntax</b>         | :SYSTem:COMMunicate:ENABle <i>mode</i> , <i>interface</i>  |  |
|                       | :SYSTem:COMMunicate:ENABle? <i>interface</i>   |  |
| <b>Parameter</b>      | <b><i>mode</i></b>   | Interface on or off. 1 ON 0 OFF. Parameter data type is boolean.                     |
|                       | <b><i>interface</i></b>  | Interface. GPIB USB LAN SOCKets TELNet VXI11 HISLip WEB. Parameter data type is CPD. |
|                       | <i>mode</i> =1 or ON enables the specified <i>interface</i> .  |  |
|                       | <i>mode</i> =0 or OFF disables the specified <i>interface</i> .  |  |
| <b>Query response</b> | <i>mode</i> <newline>  |  |
|                       | <i>mode</i> returns 0 or 1, and indicates that the specified <i>interface</i> is off or on, respectively. Response data type is NR1. |  |
| <b>Example</b>        | :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.

|                       |   |  |
|-----------------------|---|--|
| <b>Syntax</b>         | :SYSTem:COMMunicate:GPIB[:SELF]:ADDRess <i>address</i>                                |  |
|                       | :SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?  |  |
| <b>Parameter</b>      | <b><i>address</i></b>   | GPIB address, 0 to 30. Parameter data type is NR1. |
| <b>Query response</b> | <i>address</i> <newline>  |  |
|                       | <i>address</i> returns the GPIB address of the instrument. Response data type is NR1. |  |
| <b>Example</b>        | :SYST:COMM:GPIB:ADDR 17   |  |
|                       | :SYST:COMM:GPIB:ADDR?   |  |

## :SYSTem:COMMunicate:LAN:ADDRes

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.

|                       |   |   |
|-----------------------|---|---|
| <b>Syntax</b>         | :SYSTem:COMMunicate:LAN:ADDRes <i>address</i><br>:SYSTem:COMMunicate:LAN:ADDRes? [CURRENT STATic]   |   |
| <b>Parameter</b>      | <i>address</i>  | IP address of the instrument. It must be in the <i>A.B.C.D</i> format with 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 225. Parameter data type is SPD. |
| <b>Query response</b> | <i>address</i> <newline><br><br><i>address</i> returns the static LAN (IP) address of the instrument. If the CURRENT parameter is set, <i>address</i> returns the present setting. If the STATic parameter is set, <i>address</i> returns the reserved value for the next startup. Response data type is SRD. |   |
| <b>Example</b>        | :SYST:COMM:LAN:ADDR "192.168.100.100"<br>:SYST:COMM:LAN:ADDR?   |   |

## :SYSTem:COMMunicate:LAN:BSTatus?

Returns the LAN boot status of the instrument.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | :SYSTem:COMMunicate:LAN:BSTatus?   |   |
| <b>Query response</b> | <i>status</i> <newline><br><br><i>status</i> returns the following LAN boot status. Response data type is CRD. |   |
|                       | <b>LAN_AUTO_IP</b>   | The instrument booted with a local IP address.      |
|                       | <b>LAN_DHCP</b>  | The instrument booted with a DHCP-assigned address. |
|                       | <b>LAN_FAULT</b>   | The instrument cannot detect a connection.          |
|                       | <b>LAN_STATIC</b>  | The instrument booted with a static IP address.     |
| <b>Example</b>        | :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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | :SYSTem:COMMunicate:LAN:DNS[j] <i>address</i><br>:SYSTem:COMMunicate:LAN:DNS[j]? [CURRent STATic]  |   |
| <b>Parameter</b>      | <i>address</i>   | IP address of the DNS server. It must be in the <i>A.B.C.D</i> format with 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 255. Parameter data type is SPD. |
| <b>Query response</b> | <i>address</i> <newline><br><br><i>address</i> returns the IP address of the DNS server. If the CURRent parameter is set, <i>address</i> returns the present setting. If the STATic parameter is set, <i>address</i> returns the reserved value for the next startup. Response data type is SRD. |   |
| <b>Example</b>        | :SYST:COMM:LAN:DNS "192.168.100.200"<br>:SYST:COMM:LAN:DNS2?   |   |

## :SYSTem:COMMunicate:LAN:DOMain?

Returns the domain name of the network to which the instrument is connected.

|                       |   |  |
|-----------------------|---|--|
| <b>Syntax</b>         | :SYSTem:COMMunicate:LAN:DOMain?   |  |
| <b>Query response</b> | <i>domain_name</i> <newline><br><br><i>domain_name</i> returns the domain name of the network. Response data type is SRD. |  |
| <b>Example</b>        | :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.

|               |   |  |
|---------------|---|--|
| <b>Syntax</b> | :SYSTem:COMMunicate:LAN:<GATE GATeway> <i>address</i><br>:SYSTem:COMMunicate:LAN:<GATE GATeway>? [CURRent STATic] |  |
|---------------|---|--|

Subsystem Commands  
SYSTem Subsystem

**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?

Subsystem Commands  
SYSTem Subsystem

**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.

|                       |   |  |
|-----------------------|---|--|
| <b>Syntax</b>         | :SYSTem:COMMunicate:LAN:WINS[j] <i>address</i><br>:SYSTem:COMMunicate:LAN:WINS[j]? [CURRENT STATic]   |  |
| <b>Parameter</b>      | <i>address</i>  | IP address of the WINS server. It must be in the <i>A.B.C.D</i> format with 15 characters maximum. <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i> must be a number from 0 to 255. Parameter data type is SPD. |
| <b>Query response</b> | <i>address</i> <newline><br><br><i>address</i> returns the IP address of the WINS server. If the CURRENT parameter is set, <i>address</i> returns the present setting. If the STATic parameter is set, <i>address</i> returns the reserved value for the next startup. Response data type is SRD. |  |
| <b>Example</b>        | :SYST:COMM:LAN:WINS "192.168.100.150"<br>:SYST:COMM:LAN:WINS2?  |  |

## :SYSTem:DATA:QUANtity?

Returns the number of data for the specified channel in the data buffer.

|                       |  |  |
|-----------------------|--|--|
| <b>Syntax</b>         | :SYSTem:DATA:QUANtity? [ <i>chanlist</i> ]   |  |
| <b>Parameter</b>      | <i>chanlist</i>  | Channels. Parameter data type is channel list.<br>(@1)(@2)((@1,2))(@1:2)((@2,1))(@2:1). See “ <b>Channel List Parameter</b> ” on page 1-8.<br>(@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.<br>If this parameter is not specified, <i>chanlist</i> =(@1) is set. |
| <b>Query response</b> | <i>response</i> <newline><br><br><i>response</i> returns the number of data. Response data type is NR1. If both channels 1 and 2 are selected by <i>chanlist</i> , <i>response</i> returns the number of channel 1 data and the number of channel 2 data in this order. They are separated by a comma. |  |
| <b>Example</b>        | :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.

Conditions for the channels to group:

- Trigger source must be the same.
- If the trigger source is set to TIMER, the trigger period of the source output trigger (transient action) must be the same. This is not applicable if the trigger count is set to 1.

**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:LFRrequency

Selects the line frequency. This command setting is not changed by power off or the \*RST command.

**Syntax** :SYSTem:LFRrequency *frequency*  
:SYSTem:LFRrequency?

**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:PERSONa[:MANufacturer]

Changes the manufacturer name of the [\\*IDN?](#) command response.

The setting change is effective after rebooting the B2900. The manufacturer name is set back to “Keysight Technologies” by the [:SYSTem:PERSONa\[:MANufacturer\]:DEFault](#) command or by pressing the front panel System > More > Factory function keys which applies the factory shipment initial condition. This command setting is not changed by power off or the [\\*RST](#) command.

### Syntax

```
:SYSTem:PERSONa[:MANufacturer] maker
:SYSTem:PERSONa[:MANufacturer]? [maker]
```

### Parameter

***maker*** Manufacturer name, DEFault|AGILent|KEYSight|*name*.  
 Parameter data type is CPD or SPD. Query does not support *maker=name*.  
 DEFault: “Keysight Technologies”  
 AGILent: “Agilent Technologies”  
 KEYSight: “Keysight Technologies”  
*name*: a quoted string (alpha numeric and whitespace) for manufacturer name

### Query response

*maker* <newline>  
 If no parameter is specified, *maker* returns a quoted string of the present setting. If a parameter is specified, *maker* returns a quoted string associated with the parameter shown above. Response data type is SRD.

### Example

```
:SYST:PERS AGIL
:SYST:PERS?
```

## **:SYSTem:PERSonA[:MANufacturer]:DEFault**

Clears the **:SYSTem:PERSonA[:MANufacturer]** command setting and applies the default value “Keysight Technologies.”

The setting change is effective after rebooting the B2900.

**Syntax** :SYSTem:PERSonA[:MANufacturer]:DEFault

**Example** :SYST:PERS:DEF

## **:SYSTem:PERSonA:MODEl**

Changes the model number of the **\*IDN?** command response.

The setting change is effective after rebooting the B2900. The model number is set back to the default value (the model number on the front panel) by the **:SYSTem:PERSonA:MODEl:DEFault** command or by pressing the front panel System > More > Factory function keys which applies the factory shipment initial condition. This command setting is not changed by power off or the **\*RST** command.

---

**NOTE** If you change the model number, instrument control software (e.g. Quick IV, IVI-COM/C, LabVIEW, and so on) may not detect this instrument.

---

**Syntax** :SYSTem:PERSonA:MODEl *model*  
:SYSTem:PERSonA:MODEl?

**Parameter** *model* A quoted string (alpha numeric) for model number. Parameter data type is SPD.

**Query response** *response* <newline>  
*response* returns a quoted string of the present setting. Response data type is SRD.

**Example** :SYST:PERS:MOD “2912A”  
:SYST:PERS:MOD?

## **:SYSTem:PERSonA:MODEl:DEFault**

Clears the **:SYSTem:PERSonA:MODEl** command setting and applies the default value (the model number on the front panel).

The setting change is effective after rebooting the B2900.

**Syntax** :SYSTem:PERSonal:MODEl:DEFault

**Example** :SYST:PERS:MOD:DEF

## :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** :SYSTem: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?

|                  |               |   |
|------------------|---------------|---|
| <b>Parameter</b> | <i>hour</i>   | Hour. Integer from 0 to 23. Parameter data type is NR1.   |
|                  | <i>minute</i> | Minute. Integer from 0 to 59. Parameter data type is NR1. |
|                  | <i>second</i> | Second. Integer from 0 to 59. Parameter data type is NR1. |

|                       |   |
|-----------------------|---|
| <b>Query response</b> | <p><i>response</i> &lt;newline&gt;</p> <p><i>response</i> returns <i>hour, minute, second</i>. Each value is separated by a comma. Response data type is NR1.</p> |
|-----------------------|---|

**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?

|                  |             |  |
|------------------|-------------|--|
| <b>Parameter</b> | <i>mode</i> | Automatic reset function on or off. 0 OFF 1 ON (default).<br>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*]]

|                  |               |  |
|------------------|---------------|--|
| <b>Parameter</b> | <i>offset</i> | Indicates the beginning of the data received. <i>n</i>  CURRENT START (default). Parameter data type is NR1 or CPD.<br><br><i>offset=n</i> specifies the <i>n</i> +1th data. <i>n</i> is an integer, 0 to maximum (depends on the buffer state).<br><br><i>offset=CURR</i> specifies the present data position.<br><br><i>offset=STAR</i> specifies the top of trace buffer. Same as <i>offset=0</i> . |
|                  | <i>size</i>   | 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 <i>offset</i> 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]:BYPas  
s**

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]:COUN  
t**

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.

|                       |  |  |
|-----------------------|--|--|
| <b>Parameter</b>      | <b><i>arm_count</i></b>  | Arm count. <i>value</i> (1 to 100000 or 2147483647) INFinity MINimum MAXimum DEFault (default is 1). Parameter data type is NRf+. <i>value</i> =2147483647 indicates infinity.<br><br>Query does not support <i>arm_count=</i> <i>value</i> and INFinity.<br><br><i>Arm count</i> × <i>Trigger count</i> must be less than 100001. |
| <b>Query response</b> | <i>response</i> <newline>  | <i>response</i> returns the present setting of arm count. If a parameter is specified, <i>response</i> returns the value assigned to DEF, MIN, MAX, or INF. Response data type is NR1. If the arm count is set to infinity, <i>response</i> returns 2147483647.  |
| <b>Example</b>        | :ARM:COUN 10<br>:ARM2:TRAN:COUN?<br><br><b>:ARM&lt;:ACQuire :TRANsient [:ALL]&gt;[:LAYer]:DELay</b><br>Sets the arm delay for the specified device action.   |  |
| <b>Syntax</b>         | :ARM[ <i>c</i> ]<:ACQuire :TRANsient [:ALL]>[:LAYer]:DELay <i>delay</i><br>:ARM[ <i>c</i> ]<:ACQuire :TRANsient>[:LAYer]:DELay? [ <i>delay</i> ]<br>:ARM[ <i>c</i> ][:ALL][:LAYer]:DELay? <i>delay</i><br>For <:ACQuire :TRANsient [:ALL]>, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions. |  |
| <b>Parameter</b>      | <b><i>delay</i></b>  | Arm delay, in seconds. <i>value</i> (0 to 100000) MINimum MAXimum DEFault (default is 0). Parameter data type is NRf+. Query does not support <i>delay=</i> <i>value</i> .   |
| <b>Query response</b> | <i>response</i> <newline>  | <i>response</i> returns the present setting of arm delay. If a parameter is specified, <i>response</i> returns the value assigned to DEF, MIN, or MAX. Response data type is NR3.  |
| <b>Example</b>        | :ARM:DEL 0.1<br>: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.

|                       |  |   |
|-----------------------|--|---|
| <b>Syntax</b>         | <pre>:ARM[c]&lt;:ACQuire :TRANsient [:ALL]&gt;[:LAYer]:SOURce:LAN <i>lan_id</i>{,<i>lan_id</i>} :ARM[c]&lt;:ACQuire :TRANsient&gt;[:LAYer]:SOURce:LAN?</pre> <p>For &lt;:ACQuire :TRANsient [:ALL]&gt; and &lt;:ACQuire :TRANsient&gt;, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.</p> |   |
| <b>Parameter</b>      | <i>lan_id</i>  | LAN ID of the LXI trigger. LAN0 LAN1 LAN2 LAN3 LAN4 LAN5 LAN6 LAN7. All is selected as default. Parameter data type is CPD. |
| <b>Query response</b> | <p><i>response</i> &lt;newline&gt;</p> <p><i>response</i> returns the present setting, LAN0 through LAN7. Response data type is CRD. Multiple responses are separated by a comma.</p>  |   |
| <b>Example</b>        | <pre>:ARM:SOUR:LAN LAN7 :ARM2:TRAN:SOUR:LAN?</pre>   |   |

## **:ARM<:ACQuire|:TRANsient|[:ALL]>[:LAYer]:SOURce[:SIGNal]**

Selects the arm source for the specified device action.

|                  |  |   |
|------------------|--|---|
| <b>Syntax</b>    | <pre>:ARM[c]&lt;:ACQuire :TRANsient [:ALL]&gt;[:LAYer]:SOURce[:SIGNal] <i>source</i> :ARM[c]&lt;:ACQuire :TRANsient&gt;[:LAYer]:SOURce[:SIGNal]?</pre> <p>For &lt;:ACQuire :TRANsient [:ALL]&gt; and &lt;:ACQuire :TRANsient&gt;, specify :ACQuire for measurement, :TRANsient for source output, or :ALL for both device actions.</p> |   |
| <b>Parameter</b> | <i>source</i>  | Arm source. AINT (default) BUS TIme 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* (2E-5 to 1E+5 for B2901A/B2902A, 1E-5 to 1E+5 for B2911A/B2912A)|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]:TOUT  
put: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[:ALL]>:BYPass?

For <:ACQuire[:TRANsient[:ALL]> and <:ACQuire[:TRANsient[:ALL]>, 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:BYPass 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 or 2147483647)|INFinity MINimum|MAXimum|DEFault (default is 1). Parameter data type is NRf+. *value*=2147483647 indicates infinity.  
Query does not support *trigger\_count=value* and INFinity.  
*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 100000)|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.

## Subsystem Commands

### TRIGger Subsystem

*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* (2E-5 to 1E+5 for B2901A/B2902A, 1E-5 to 1E+5 for B2911A/B2912A)|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:TOU:SIGN EXT3
:TRIG2:TRAN:TOU: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]?
```

## Subsystem Commands

### TRIGger Subsystem

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**

## Error Messages

This chapter shows the error code/messages returned from Keysight 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 Keysight 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 -1XX 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.

|                   |   |
|-------------------|---|
| <b>Error -111</b> | <p>Header separator error</p> <p>An illegal character for a header separator was received; for example, no white space between the header and parameter.</p>  |
| <b>Error -112</b> | <p>Program mnemonic too long</p> <p>A keyword in the command header contains more than twelve characters.</p>   |
| <b>Error -113</b> | <p>Undefined header</p> <p>An undefined command header was received; for example, *XYZ.</p>   |
| <b>Error -114</b> | <p>Header suffix out of range</p> <p>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.</p>              |
| <b>Error -120</b> | <p>Numeric data error</p> <p>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.</p> |
| <b>Error -121</b> | <p>Invalid character in number</p> <p>An invalid character for the data type was received; for example, an alpha-character was received when the type was decimal numeric.</p>                        |
| <b>Error -123</b> | <p>Exponent too large</p> <p>The magnitude of the exponent was larger than 32000.</p>   |
| <b>Error -124</b> | <p>Too many digits</p> <p>The mantissa of a decimal numeric data contained more than 255 digits excluding leading zeros.</p>  |
| <b>Error -128</b> | <p>Numeric data not allowed</p> <p>Numeric data is not allowed in this position for this command.</p>   |
| <b>Error -130</b> | <p>Suffix error</p> <p>An error was detected in the suffix. This error message is reported if B2900 cannot determine the more specific suffix errors -131 through -138.</p>                           |
| <b>Error -131</b> | <p>Invalid suffix</p>   |

Error Messages  
Command Error

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.



|                   |   |
|-------------------|---|
| <b>Error -161</b> | <p>Invalid block data</p> <p>An invalid block data was received; for example, an END message was received before the length was satisfied.</p>              |
| <b>Error -168</b> | <p>Block data not allowed</p> <p>A legal block data was received but was not allowed at this point.</p>   |
| <b>Error -170</b> | <p>Expression error</p> <p>An error was detected in an expression. This error is reported if B2900 cannot determine more specific errors -171 and -178.</p> |
| <b>Error -171</b> | <p>Invalid expression</p> <p>The expression was invalid; for example, unmatched parentheses or an illegal character.</p>                                    |
| <b>Error -178</b> | <p>Expression data not allowed</p> <p>An expression was received but was not allowed at this point.</p>   |

## Execution Error

Keysight B2900 reports -2XX errors when it is unable to perform a valid programming command.

|                   |   |
|-------------------|---|
| <b>Error -200</b> | Execution error<br>Generic execution error for B2900 that cannot be determined more specifically.   |
| <b>Error -220</b> | Parameter error; Invalid channel list<br>Parameter error; Invalid group definition<br>Invalid channel list or group definition was specified. Set appropriate value.  |
| <b>Error -221</b> | Settings conflict; <i>message</i> ; channel <i>n</i><br>A specified parameter setting could not be executed due to the present device state. Check the settings specified by <i>message</i> and channel <i>n</i> , and set appropriate value. |
| <b>Error -222</b> | Data out of range; <i>message</i> ; channel <i>n</i><br>Interpreted value of the program was out of range as defined by B2900. Check the B2900 settings specified by <i>message</i> and channel <i>n</i> , and set appropriate value.         |
| <b>Error -223</b> | Too much data<br>Too many parameters were sent. Reduce number of list data.   |
| <b>Error -224</b> | Illegal parameter value<br>Illegal parameter value was sent. Set appropriate parameter value.   |
| <b>Error -230</b> | Data corrupt or stale<br>Possibly invalid data; new reading started but not completed since last access.  |
| <b>Error -231</b> | Data questionable<br>Measurement accuracy is suspect.   |
| <b>Error -232</b> | Invalid format<br>The data format or structure is inappropriate.  |

|                   |   |
|-------------------|---|
| <b>Error -233</b> | <p>Invalid version</p> <p>The version of the data format is incorrect to the instrument.</p>  |
| <b>Error -240</b> | <p>Hardware error</p> <p>A hardware problem in B2900. This error message is reported if B2900 cannot detect the more specific error -241.</p>   |
| <b>Error -241</b> | <p>Hardware missing; To recover channel, execute *TST?</p> <p>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.</p> |

## Device-Dependent Error

-3XX errors indicate that Keysight 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.

|                   |   |
|-------------------|---|
| <b>Error -300</b> | Device-specific error<br><br>Generic device-dependent error for B2900 that cannot be determined more specifically.  |
| <b>Error -310</b> | System error<br><br>Some error, termed “system error” by B2900, has occurred.   |
| <b>Error -311</b> | Memory error<br><br>An error was detected in B2900’s memory.  |
| <b>Error -313</b> | Calibration memory lost; Calibration data has been lost,<br>Calibration data is initialized; Channel <i>n</i><br><br>Calibration memory lost; Nonvolatile data saved by the<br>*CAL? command has been lost; Channel <i>n</i><br><br>Non-volatile data related to the *CAL? command has been lost. |
| <b>Error -315</b> | Configuration memory lost<br><br>Non-volatile configuration data saved by B2900 has been lost.  |
| <b>Error -321</b> | Out of memory<br><br>Too many data was sent at a time.  |
| <b>Error -350</b> | Queue overflow<br><br>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 Keysight 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 Keysight B2900 specific errors, not standard SCPI errors. Consult service for errors 111 to 140.

|                  |   |
|------------------|---|
| <b>Error 101</b> | Wrong password  |
| <b>Error 102</b> | Enter password for calibration  |
| <b>Error 103</b> | Data load failed  |
| <b>Error 104</b> | Data save failed  |
| <b>Error 111</b> | Self-calibration failed; Voltage offset<br>Failed the voltage offset self-calibration specified by <i>item</i> and channel <i>n</i> .                                 |
| <b>Error 112</b> | Self-calibration failed; Current offset, <i>item</i> ; channel <i>n</i><br>Failed the current offset self-calibration specified by <i>item</i> and channel <i>n</i> . |
| <b>Error 113</b> | Self-calibration failed; Voltage gain, <i>item</i> ; channel <i>n</i><br>Failed the voltage gain self-calibration specified by <i>item</i> and channel <i>n</i> .     |
| <b>Error 114</b> | Self-calibration failed; Current gain, <i>item</i> ; channel <i>n</i><br>Failed the current gain self-calibration specified by <i>item</i> and channel <i>n</i> .     |
| <b>Error 115</b> | Self-calibration failed; CMR DAC, <i>item</i> ; channel <i>n</i><br>Failed the CMR DAC self-calibration specified by <i>item</i> and channel <i>n</i> .               |
| <b>Error 121</b> | Self-test failed; CPU communication, <i>item</i> ; channel <i>n</i><br>Failed the CPU communication test specified by <i>item</i> and channel <i>n</i> .              |
| <b>Error 122</b> | Self-test failed; Fan status, <i>item</i> ; channel <i>n</i><br>Failed the fan status test specified by <i>item</i> and channel <i>n</i> .                            |
| <b>Error 131</b> | Self-test failed; SMU communication, <i>item</i> ; channel <i>n</i><br>Failed the SMU communication test specified by <i>item</i> and channel <i>n</i> .              |

|                  |   |
|------------------|---|
| <b>Error 132</b> | Self-test failed; CPLD access, <i>item</i> ; channel <i>n</i><br>Failed the CPLD access test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 133</b> | Self-test failed; Trigger count, <i>item</i> ; channel <i>n</i><br>Failed the trigger count test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 134</b> | Self-test failed; DAC/ADC, <i>item</i> ; channel <i>n</i><br>Failed the DAC/ADC test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 135</b> | Self-test failed; Loop control, <i>item</i> ; channel <i>n</i><br>Failed the loop control test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 136</b> | Self-test failed; I sense, <i>item</i> ; channel <i>n</i><br>Failed the current sense test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 137</b> | Self-test failed; V sense, <i>item</i> ; channel <i>n</i><br>Failed the voltage sense test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 138</b> | Self-test failed; F-COM comparison, <i>item</i> ; channel <i>n</i><br>Failed the F-COM comparison test specified by <i>item</i> and channel <i>n</i> .                                    |
| <b>Error 139</b> | Self-test failed; V switch, <i>item</i> ; channel <i>n</i><br>Failed the voltage switch test specified by <i>item</i> and channel <i>n</i> .  |
| <b>Error 140</b> | Self-test failed; Temperature sensor, <i>item</i> ; channel <i>n</i><br>Failed the temperature sensor test specified by <i>item</i> and channel <i>n</i> .                                |
| <b>Error 141</b> | Self-test skipped; To recover channel, execute *TST?  |
| <b>Error 201</b> | Not able to perform requested operation   |
| <b>Error 202</b> | Not allowed; Instrument locked by another I/O session<br>The requested operation is not allowed because the instrument is locked by another I/O session. The instrument must be unlocked. |
| <b>Error 203</b> | Not able to execute while instrument is measuring   |

Error Messages  
B2900 Specific Error

|                  |  |
|------------------|--|
| <b>Error 210</b> | Operation is not completed<br>Operation is still in progress. Wait for operation complete.   |
| <b>Error 211</b> | Cannot switch low sense terminal with output on<br>Output relay must be off to switch low sense terminal.  |
| <b>Error 212</b> | Output relay must be on<br>Output relay must be on to perform the requested operation.   |
| <b>Error 213</b> | Output relay must be off<br>Output relay must be off to perform the requested operation.   |
| <b>Error 214</b> | Display must be enabled<br>Display is currently disabled. Set remote display on.   |
| <b>Error 215</b> | Remote sensing must be on<br>Remote sensing must be on to perform the requested operation.   |
| <b>Error 216</b> | Auto resistance measurement must be off<br>Automatic resistance measurement must be off to perform the requested operation.  |
| <b>Error 290</b> | Not able to recall state: it is empty  |
| <b>Error 291</b> | State file size error  |
| <b>Error 292</b> | State file corrupt   |
| <b>Error 301</b> | Emergency; Overvoltage status detected; Channel <i>n</i><br>Overvoltage status was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.                 |
| <b>Error 302</b> | Emergency; Overcurrent status (245 V) detected; Channel <i>n</i><br>overcurrent status (245 V) was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command. |
| <b>Error 303</b> | Emergency; Overcurrent status (35 V) detected; Channel <i>n</i>  |



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 Messages  
B2900 Specific Error

|                  |  |
|------------------|--|
| <b>Error 315</b> | Emergency; Low sense (plus) abuse detected; Channel <i>n</i><br><br>Low sense (plus) abuse status was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.                |
| <b>Error 321</b> | Emergency; SMU main power supply failure detected; Channel <i>n</i><br><br>SMU main power supply failure was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.         |
| <b>Error 322</b> | Emergency; SMU positive power supply failure detected; Channel <i>n</i><br><br>SMU positive power supply failure was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command. |
| <b>Error 323</b> | Emergency; SMU negative power supply failure detected; Channel <i>n</i><br><br>SMU negative power supply failure was detected in channel <i>n</i> . All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command. |
| <b>Error 324</b> | Emergency; SMU power supply was turned off; Channel <i>n</i><br><br>SMU power supply was turned off because emergency status was detected in channel <i>n</i> . All channels were disabled. Execute the *TST? command.                                   |
| <b>Error 331</b> | Emergency; Interlock open detected<br><br>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.            |
| <b>Error 341</b> | Emergency; Fan speed is too slow<br><br>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.   |
| <b>Error 342</b> | Emergency; Fan speed is too fast<br><br>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.   |
| <b>Error 351</b> | Emergency; Internal communication failure detected by SMU; Channel <i>n</i>  |

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 Messages  
B2900 Specific Error

|                  |  |
|------------------|--|
| <b>Error 364</b> | Emergency; Unexpected sense data was received; Channel <i>n</i><br><br>Unexpected sense data was received. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.  |
| <b>Error 365</b> | Emergency; Sense data was not received in Timer period; Channel <i>n</i><br><br>Data communication failure. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command. |
| <b>Error 366</b> | Emergency; Timestamp FIFO overflow detected; Channel <i>n</i><br><br>Data communication failure. All channel output is changed to 0 V and the all output switch is opened. Execute the *TST? command.            |
| <b>Error 600</b> | Some or all licenses from license file(s) could not be installed   |
| <b>Error 700</b> | ProgramMemory; Program size overflow<br><br>Program memory cannot save the program. Reduce program size.   |
| <b>Error 701</b> | ProgramMemory; Invalid variable<br><br>Appropriate variable name must be specified.  |
| <b>Error 702</b> | ProgramMemory; Invalid variable number<br><br>Appropriate variable name must be specified.   |
| <b>Error 703</b> | ProgramMemory; Query command is not supported<br><br>Memory program cannot contain query command.  |
| <b>Error 704</b> | ProgramMemory; Program is not selected<br><br>Appropriate program name must be specified.  |
| <b>Error 705</b> | ProgramMemory; Cannot execute program while another program is running<br><br>Another program is running. Execute the program after it is stopped.   |
| <b>Error 706</b> | 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

Error Messages  
B2900 Specific Error

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; Mismatched 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.

|                  |   |
|------------------|---|
| <b>Error 816</b> | Calculate; Invalid hex format string is used<br>Data contains invalid hex format string. Enter appropriate expression.                  |
| <b>Error 817</b> | Calculate; Invalid channel number is used<br>Expression contains invalid channel number. Enter appropriate expression.                  |
| <b>Error 818</b> | Calculate; Null expression<br>Expression is not defined. Enter appropriate expression.  |
| <b>Error 819</b> | Calculate; Null expression in parentheses<br>Expression contains empty parentheses. Enter appropriate expression.                       |
| <b>Error 820</b> | Calculate; Null expression in brackets<br>Expression contains empty brackets. Enter appropriate expression.                             |
| <b>Error 821</b> | Calculate; Fed disabled MATH for limit test<br>Limit test tried to feed the math result currently disabled. Enable the math expression. |
| <b>Error 822</b> | Calculate; Mismatched trigger counts<br>Trigger count of grouped channels must be the same.   |
| <b>Error 823</b> | Calculate; Mismatched vector lengths<br>Vector length of grouped channels must be the same.   |
| <b>Error 824</b> | Calculate; Invalid character in math name<br>Appropriate math expression name must be specified.  |
| <b>Error 861</b> | Trace; Illegal with storage active<br>Storage device must be idle to perform the requested operation.                                   |
| <b>Error 862</b> | Trace; No trace data<br>Trace buffer must contain data to perform the requested operation.  |
| <b>Error 870</b> | Macro file size error<br>Macro file size error. Reduce file size.   |

Error Messages  
B2900 Specific Error

|                  |   |
|------------------|---|
| <b>Error 871</b> | Cannot create state data on non-volatile memory   |
| <b>Error 900</b> | Internal system error   |
| <b>Error 950</b> | Unsupported parameter<br>Conventional command set error. Specified parameter is not supported by B2900. |
| <b>Error 951</b> | Unsupported command<br>Conventional command set error. Specified command is not supported by B2900.     |





Keysight 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]

Using Your Existing Programs  
Conventional commands supported by B2900

: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>

Using Your Existing Programs  
Conventional commands supported by B2900

: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|SREa|[,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]

Using Your Existing Programs  
Conventional commands supported by B2900

[[: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?

Using Your Existing Programs  
Conventional commands supported by B2900

: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>

Using Your Existing Programs  
Conventional commands supported by B2900

: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)

Using Your Existing Programs  
Conventional commands partially supported by B2900

: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]]:VOLTag[:DC]:PROTection:RSYNchronize <Bool>  
:SOURce[1]:CLEar:AUTO:MODE <ALWays|TCOut>  
:SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACTor <NRf>  
:SOURce[1]:CURRent[:LEVel]:TRIGgered:SFACTor:STATe <Bool>  
:SOURce[1]:FUNCTION[:MODE] <VOLTag|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]:VOLTag[:LEVel]:TRIGgered:SFACTor <NRf>  
:SOURce[1]:VOLTag[:LEVel]:TRIGgered:SFACTor:STATe <Bool>  
:STATus:QUEue:DISable (NR1 list)

Using Your Existing Programs  
Conventional commands not supported by B2900

: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>



This information is subject to change without notice.

© Keysight Technologies 2011-2019

Edition 8, March 2019



B2910-90030  
[www.keysight.com](http://www.keysight.com)