

# DC Programmable Electronic Loads

## Series IT8700 Programming Guide



Model: IT8722/IT8723/IT8731/IT8732/IT8733/  
IT8722B/IT8732B/IT8733B/  
IT8722P/IT8723P/IT8731P/IT8732P/IT8733P/  
IT8722BP/IT8732BP/IT8733BP/IT8733BP+/  
IT8721P+/IT8731P+/IT8722P+/IT8723P+/  
IT8732P+/IT8733P+/IT8722BP+/IT8732BP+/  
IT8702/IT8703/IT8701P/IT8702P/IT8703P  
Version: 2.6

## Notices

© Itech Electronic, Co., Ltd. 2024  
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 permission and written consent from Itech Electronic, Co., Ltd. as governed by international copyright laws.

### Manual Part Number

IT8700-402216

### Revision

Second Edition, Apr. 19,  
2024

Itech Electronic, Co., Ltd.

### Trademarks

Pentium is U.S. registered trademarks of Intel Corporation.

Microsoft, Visual Studio, Windows and MS Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries and regions.

## Warranty

The materials contained in this document are provided “as is”, and is subject to change, without prior notice, in future editions. Further, to the maximum extent permitted by applicable laws, ITECH 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. ITECH shall not be held liable for errors or for incidental or indirect damages in connection with the furnishing, use or application of this document or of any information contained herein. Should ITECH and the user enter into a separate written agreement with warranty terms covering the materials in this document that conflict with these terms, the warranty terms in the separate agreement shall prevail.

### Technology Licenses

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

### Restricted Rights Legend

Restricted permissions of the U.S. government. Permissions for software and technical data which are authorized to the U.S. Government only include those for custom provision to end users. ITECH follows FAR 12.211 (technical data), 12.212 (computer software). DFARS 252.227-7015 (technical data--commercial products) for national defense and DFARS 227.7202-3 (permissions for commercial computer software or computer software documents) while providing the customized business licenses of software and technical data.

## Safety Notices

### CAUTION

A CAUTION sign denotes a hazard. It calls attention to an operating procedure or practice that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

### WARNING

A WARNING sign denotes a hazard. It calls attention to an operating procedure or practice that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.



### NOTE

A NOTE sign denotes important hint. It calls attention to tips or supplementary information that is essential for users to refer to.

## Quality Certification and Assurance

We certify that series IT8700 electronic load meets all the published specifications at time of shipment from the factory.

## Warranty

ITECH warrants that the product will be free from defects in material and workmanship under normal use for a period of one (1) year from the date of delivery (except those described in the Limitation of Warranty below).

For warranty service or repair, the product must be returned to a service center designated by ITECH.









- The product returned to ITECH for warranty service must be shipped PREPAID. And ITECH will pay for return of the product to customer.
- If the product is returned to ITECH for warranty service from overseas, all the freights, duties and other taxes shall be on the account of customer.








## Limitation of Warranty

This Warranty will be rendered invalid in case of the following:

- Damage caused by circuit installed by customer or using customer own products or accessories;
- Modified or repaired by customer without authorization;
- Damage caused by circuit installed by customer or not operating our products under designated environment;
- The product model or serial number is altered, deleted, removed or made illegible by customer;
- Damaged as a result of accidents, including but not limited to lightning, moisture, fire, improper use or negligence.

## Safety Symbols

	Direct current		ON (power on)
	Alternating current		OFF (power off)
	Both direct and alternating current		Power-on state
	Protective conductor terminal		Power-off state

	Earth (ground) terminal		Reference terminal
	Caution, risk of electric shock		Positive terminal
	Warning, risk of danger (refer to this manual for specific Warning or Caution information)		Negative terminal
	Frame or chassis terminal	-	-

## Safety Precautions

The following safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or specific warnings elsewhere in this manual will constitute a default under safety standards of design, manufacture and intended use of the instrument. ITECH assumes no liability for the customer's failure to comply with these precautions.

### WARNING

- Do not use the instrument if it is damaged. Before operation, check the casing to see whether it cracks. Do not operate the instrument in the presence of inflammable gasses, vapors or dusts.
- The electronic load is provided with a three-core power line during delivery and should be connected to a three-core junction box. Before operation, be sure that the instrument is well grounded.
- Make sure to use the power cord supplied by ITECH.
- Check all marks on the instrument before connecting the instrument to power supply.
- Use electric wires of appropriate load. All loading wires should be capable of bearing maximum short-circuit current of electronic load without overheating. If there are multiple electronic loads, each pair of the power cord must be capable of bearing the full-loaded rated short-circuit output current
- Ensure the voltage fluctuation of mains supply is less than 10% of the working voltage range in order to reduce risks of fire and electric shock.
- Do not install alternative parts on the instrument or perform any unauthorized modification.
- Do not use the instrument if the detachable cover is removed or loosen.
- To prevent the possibility of accidental injuries, be sure to use the power adapter supplied by the manufacturer only.
- We do not accept responsibility for any direct or indirect financial damage or loss of profit that might occur when using the instrument.
- This instrument is used for industrial purposes, do not apply this product to IT power supply system.
- Never use the instrument with a life-support system or any other equipment subject to safety requirements.

**CAUTION**

- Failure to use the instrument as directed by the manufacturer may render its protective features void.
- Always clean the casing with a dry cloth. Do not clean the internals.
- Make sure the vent hole is always unblocked.

## Environmental Conditions



The instrument is designed for indoor use and an area with low condensation. The table below shows the general environmental requirements for the instrument. The speed of fan will change intelligently by the temperature of radiator. When the temperature is up to 40°C, the fan will be on and adjust intelligently when temperature changes.

Environmental Conditions	Requirements
Operating temperature	0°C to 40°C
Operating humidity	20%-80% (non-condensation)
Storage temperature	-20°C to 70 °C
Altitude	Operating up to 2,000 meters
Pollution degree	Pollution degree 2
Installation category	II


**Note**

To make accurate measurements, allow the instrument to warm up for 30 min before operation.

## Regulatory Markings

	The CE mark indicates that the product complies with all the relevant European legal directives. The specific year (if any) affixed refers to the year when the design was approved.
	The instrument complies with the WEEE Directive (2002/96/EC) marking requirement. This affixed product label indicates that you must not discard the electrical/electronic product in domestic household waste.



This symbol indicates the time period during which no hazardous or toxic substances are expected to leak or deteriorate during normal use. The expected service life of the product is 10 years. The product can be used safely during the 10-year Environment Friendly Use Period (EFUP). Upon expiration of the EFUP, the product must be immediately recycled.

## Waste Electrical and Electronic Equipment (WEEE) Directive



2002/96/EC Waste Electrical and Electronic Equipment (WEEE) Directive

This product complies with the WEEE Directive (2002/96/EC) marking requirement. This affix product label indicates that you must not discard the electrical/electronic product in domestic household waste.

Product Category

With reference to the equipment classifications described in the Annex I of the WEEE Directive, this instrument is classified as a "Monitoring and Control Instrument".

To return this unwanted instrument, contact your nearest ITECH office.

## Compliance Information

Complies with the essential requirements of the following applicable European Directives, and carries the CE marking accordingly:

- Electromagnetic Compatibility (EMC) Directive 2014/30/EU
- Low-Voltage Directive (Safety) 2014/35/EU

Conforms with the following product standards:

### EMC Standard

IEC 61326-1:2012/ EN 61326-1:2013 <sup>123</sup>

#### Reference Standards

CISPR 11:2009+A1:2010/ EN 55011:2009+A1:2010 (Group 1, Class A)

IEC 61000-4-2:2008/ EN 61000-4-2:2009

IEC 61000-4-3:2006+A1:2007+A2:2010/ EN 61000-4-3:2006+A1:2008+A2:2010

IEC 61000-4-4:2004+A1:2010/ EN 61000-4-4:2004+A1:2010

IEC 61000-4-5:2005/ EN 61000-4-5:2006

IEC 61000-4-6:2008/ EN 61000-4-6:2009

IEC 61000-4-11:2004/ EN 61000-4-11:2004

1. The product is intended for use in non-residential/non-domestic environments. Use of the product in residential/domestic environments may cause electromagnetic interference.
2. Connection of the instrument to a test object may produce radiations beyond the specified limit.
3. Use high-performance shielded interface cable to ensure conformity with the EMC standards listed above.

### Safety Standard

IEC 61010-1:2010/ EN 61010-1:2010

## Content

Quality Certification and Assurance .....	i
Warranty.....	i
Limitation of Warranty .....	i
Safety Symbols .....	i
Safety Precautions.....	ii
Environmental Conditions .....	iii
Regulatory Markings .....	iii
Waste Electrical and Electronic Equipment (WEEE) Directive.....	iv
Compliance Information.....	v
<b>Chapter1 Remote Control .....</b>	<b>1</b>
1.1 Communication Interfaces .....	1
1.1.1 GPIB Capabilities of the Electronic Load .....	1
1.1.2 RS-232 Capabilities of the Electronic Load .....	2
1.1.3 USB-TMC Capabilities of the Electronic Load .....	4
1.2 Programming the Status Registers .....	4
1.3 Condition registers .....	7
1.4 Event registers .....	8
1.5 Enable registers .....	8
1.6 Queues .....	9
1.6.1 Output queue.....	9
1.6.2 Error queue.....	9
1.7 Status Byte and Service Request (SRQ) .....	9
1.7.1 Status Byte Register.....	9
1.7.2 Service request enable register .....	10
1.8 Serial poll and SRQ .....	11
1.9 Trigger model (GPIB operation).....	11
1.9.1 Idle and initiate .....	12
1.9.2 Trigger model operation.....	12
<b>Chapter2 SCPI Introduction.....</b>	<b>14</b>
2.1 SCPI Introduction.....	14
2.2 Types of SCPI Commands .....	14
2.2.1 Multiple Commands in a Message.....	15
2.2.2 Moving Among Subsystems.....	15
2.2.3 Including Common Commands .....	16
2.2.4 Case sensitivity .....	16
2.2.5 Long-form and short-form versions.....	16
2.2.6 Using Queries.....	17
2.3 Types of SCPI Messages .....	17
2.3.1 The Message Unit.....	17
2.3.2 Headers.....	18
2.3.4 Query Indicator.....	18
2.3.5 Message Unit Separator.....	18
2.3.6 Root Specifier .....	18
2.3.7 Message Terminator.....	18
2.3.8 Command execution rules.....	18



2.4 SCPI Data Formats .....	19
2.5 Response Data Types .....	20
2.6 SCPI Command Completion.....	21
2.7 SCPI Conformance Information .....	22
2.7.1 Language Dictionary Introduction .....	22
2.8 Subsystem Commands .....	23
Chapter3    Programming Examples .....	24
Example 1: Identifying the Load in Use .....	24
Example 2: Common input commands .....	24
Example 3: Programming Transients .....	25
Example 4: Programming Lists.....	26
Example 5: Trace function .....	27
Chapter4    IEEE-488 Command Reference .....	28
*CLS — Clear Status.....	29
*ESE <NRf> — Event Enable .....	29
*ESR? .....	30
*IDN? .....	31
*RDT? .....	31
*OPC.....	31
*PSC.....	32
*RCL.....	33
*RST .....	33
*SAV.....	35
*SRE.....	35
*STB? .....	36
*TRG .....	37
*TST? .....	37
*WAI .....	38
Chapter5    System Commands .....	39
SYSTem:PRESet .....	39
SYSTem:POSetup .....	39
SYSTem:VERSion? .....	40
SYSTem:ERRor? .....	40
SYSTem:CLEar .....	41
SYSTem:LOCal .....	41
SYSTem:REMote.....	41
SYSTem:RWLock .....	42
Chapter6    STATus Subsystem .....	43
STATus:CHANnel?.....	43
STATus:CHANnel:CONDition? .....	43
STATus:CHANnel:ENABle.....	44
STATus:CSUMary:EVENT? .....	44
STATus:CSUMary:ENABle.....	45
STATus:OPERation? .....	45
STATus:OPERation:CONDition?.....	46
STATus:OPERation:ENABle .....	46

STATus:QUEStionable?	47
STATus:QUEStionable:CONDition?	48
STATus:QUEStionable:ENABle	48
STATus:PRESet	49
Chapter7    SCPI Measurement Command	50
FETCh:VOLTage[:DC]?	50
MEASure:VOLTage[:DC]?	50
FETCh:VOLTage:MAX?	50
MEASure:VOLTage:MAX?	50
FETCh:VOLTage:MIN?	51
MEASure:VOLTage:MIN?	51
FETCh:CURRent[:DC]?	51
MEASure:CURRent[:DC]?	51
FETCh:CURRent:MAX?	52
MEASure:CURRent:MAX?	52
FETCh:CURRent:MIN?	52
MEASure:CURRent:MIN?	52
FETCh:POWer[:DC]?	53
FETCh:CAPability?	53
MEASure:CAPability?	53
FETCh:ALLVoltage?	53
MEASure:ALLVoltage?	53
FETCh:ALLCurrent?	54
MEASure:ALLCurrent?	54
MEASure:ALLPower?	54
Chapter8    CHANnel Subsystem	56
CHANnel	56
CHANnel:ID?	56
Chapter9    TRACe Subsystem	58
TRACe:CLEar	58
TRACe:FREE?	58
TRACe:POINts	58
TRACe:FEED	59
TRACe:FEED:CONTRol	60
TRACe:DATA?	60
TRACe:FILTer	61
TRACe:DELAy	61
TRACe:TIMer	62
Chapter10   SOURce Subsystem	63
[SOURce:]INPut:ALL	63
[SOURce:]INPut	63
[SOURce:]INPut:CONTRol	64
[SOURce:]INPut:SYNCon	64
[SOURce:]INPut:SHORt	65
[SOURce:]INPut:TIMer	65
[SOURce:]INPut:TIMer:DELAy	66

[SOURce:]REMOte:SENSe .....	67
[SOURce:]FUNctIon .....	67
[SOURce:]FUNctIon:MODE .....	68
[SOURce:]TRANsient.....	68
[SOURce:]PROTection:CLEar .....	69
[SOURce:]CURRent .....	69
[SOURce:]CURRent:RANGe .....	70
[SOURce:]CURRent:SLEW .....	71
[SOURce:]CURRent:SLEW:POSitive.....	72
[SOURce:]CURRent:SLEW:NEGative .....	72
[SOURce:]CURRent:PROTection[:STATe] .....	73
[SOURce:]CURRent:PROTection:LEVel .....	74
[SOURce:]CURRent:PROTection:DELay .....	75
[SOURce:]CURRent:TRANsient:MODE.....	75
[SOURce:]CURRent:TRANsient:ALEVel .....	76
[SOURce:]CURRent:TRANsient:BLEVel .....	76
[SOURce:]CURRent:TRANsient:AWIDth.....	77
[SOURce:]CURRent:TRANsient:BWIDth.....	77
[SOURce:]CURRent:HIGH.....	78
[SOURce:]CURRent:LOW .....	78
[SOURce:]CURRent:TRIGgered .....	78
[SOURce:]VOLTage.....	79
[SOURce:]VOLTage:RANGe .....	80
[SOURce:]VOLTage:RANGe:AUTO[:STATe] .....	80
[SOURce:]VOLTage:ON.....	81
[SOURce:]VOLTage:LATCh .....	82
[SOURce:]VOLTage:HIGH .....	82
[SOURce:]VOLTage:LOW .....	82
[SOURce:]VOLTage:ILIMit <NRf+>.....	83
[SOURce:]VOLTage[:LOOP]:RATE <SLOW   FAST> .....	83
[SOURce:]VOLTage:TRIGgered .....	84
[SOURce:]RESistance .....	84
[SOURce:]RESistance:RANGe.....	85
[SOURce:]RESistance:HIGH.....	86
[SOURce:]RESistance:LOW .....	86
[SOURce:]RESistance:ILIMit <NRf+> .....	86
[SOURce:]RESistance:VDRop .....	87
[SOURce:]RESistance:LED[:STATe].....	88
[SOURce:]POWER .....	88
[SOURce:]POWER:RANGe .....	89
[SOURce:]POWER:HIGH .....	90
[SOURce:]POWER:LOW .....	90
[SOURce:]POWER:ILIMit <NRf+>.....	90
[SOURce:]POWER:PROTection[:LEVel] .....	91
[SOURce:]POWER:PROTection:DELay.....	92
[SOURce:]POWER:CONFig.....	92

[SOURce:]POWer:TRIGgered .....	93
[SOURce:]LIST:RANGE.....	94
[SOURce:]LIST:COUNT.....	94
[SOURce:]LIST:STEP .....	95
[SOURce:]LIST:LEVel.....	95
[SOURce:]LIST:SLEW .....	96
[SOURce:]LIST:WIDth.....	97
[SOURce:]LIST:SAV .....	97
[SOURce:]LIST:RCL .....	98
Chapter11 SENSE subsystem.....	99
SENSE:AVERAge:COUNT.....	99
SENSE:ACQuire:LINE:FREQ <NR1>.....	99
SENSE:ACQuire:NPLCount <NR1>.....	100
Chapter12 OCP Testing Commands.....	101
[SOURce:]OCP:STATe <bool>.....	101
[SOURce:]OCP:VON .....	101
[SOURce:]OCP:DELay .....	102
[SOURce:]OCP[:CURRent]:RANGE .....	102
[SOURce:]OCP[:CURRent]:STARt.....	103
[SOURce:]OCP[:CURRent]:END.....	103
[SOURce:]OCP[:CURRent]:INCrement .....	103
[SOURce:]OCP[:CURRent]:WIDTh.....	104
[SOURce:]OCP:VOLTAge:TRIP.....	104
[SOURce:]OCP[:CURRent]:LIMit[:HIGH] .....	105
[SOURce:]OCP[:CURRent]:LIMit:LOW.....	105
[SOURce:]OCP:SAVe.....	106
[SOURce:]OCP:RECall.....	106
[SOURce:]OCP:RESult? .....	107
[SOURce:]OCP:RESult:CURRent? .....	107
Chapter13 OPP Testing Commands.....	108
[SOURce:]OPP:STATe <bool>.....	108
[SOURce:]OPP:VON .....	108
[SOURce:]OPP:DELay .....	109
[SOURce:]OPP[:CURRent]:RANGE .....	109
[SOURce:]OPP[:POWer]:STARt .....	110
[SOURce:]OPP[:POWer]:END.....	110
[SOURce:]OPP[:POWer]:INCrement .....	110
[SOURce:]OPP[:POWer]:WIDTh.....	111
[SOURce:]OPP:VOLTAge:TRIP.....	111
[SOURce:]OPP[:POWer]:LIMit[:HIGH] .....	112
[SOURce:]OPP[:POWer]:LIMit:LOW.....	112
[SOURce:]OPP:SAVe.....	113
[SOURce:]OPP:RECall.....	113
[SOURce:]OPP:RESult? .....	114
[SOURce:]OPP:RESult:POWer? .....	114
Chapter14 Error Messages .....	115

# Chapter1 Remote Control

## 1.1 Communication Interfaces

### 1.1.1 GPIB Capabilities of the Electronic Load

All electronic load functions except for setting the communication parameters are programmable over the GPIB. The IEEE 488.2 capabilities of the electronic load are described in table below.

**Table IEEE 488 Capabilities of Electronic Load**

GPIB Capabilities	Response	Interface Function
Talker/Listener	All electronic load functions except for setting the communication parameters are programmable over the GPIB. The electronic load can send and receive messages over the GPIB. Status information is sent using a serial poll.	AH1, SH1, T6, L4
Service Request	The electronic load sets the SRQ line true if there is an enabled service request condition.	SR1
Remote/Local	In local mode, the electronic load is controlled from the front panel but will also execute commands sent over the GPIB. The electronic load powers up in local mode and remains in local mode until it receives a command over the GPIB. Once the electronic load is in remote mode the front panel REM annunciator is on, all front panel keys (except <b>&lt;Shift&gt; + &lt;Local&gt;</b> ) are disabled, and the display is in normal metering mode. Pressing <b>&lt;Shift&gt; + &lt;Local&gt;</b> on the front panel returns the electronic load to local mode. can be disabled using local lockout so that only the controller or the power switch can return the electronic load to local mode.	RL1
Device Trigger	The electronic load will respond to the device trigger function.	DT1
Group Execute Trigger	The electronic load will respond to the group execute trigger function.	GET
Device Clear	The electronic load responds to the Device Clear ( <b>DCL</b> ) and Selected Device Clear ( <b>SDC</b> ) interface commands. They cause the electronic load to clear any activity that would prevent it from receiving and executing a new command (including <b>*WAI</b> and <b>*OPC?</b> ). <b>DCL</b> and <b>SDC</b> do not change any programmed settings.	DCL,SDC

### GPIB Address

The electronic load operates from a GPIB address that is set from the front panel. To set the GPIB address, press the **<Shift> + <System>** key on the front panel and enter the address using the Entry keys. The address can be set from 0 to 31. The GPIB address is stored in non-volatile memory.

## 1.1.2 RS-232 Capabilities of the Electronic Load

The electronic load provides an RS-232 programming interface, which is activated by commands located under the front panel <Shift> + <System> key. All SCPI commands are available through RS-232 programming. When the RS-232 interface is selected, The EIA RS-232 Standard defines the interconnections between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). The electronic load is designed to be a DTE. It can be connected to another DTE such as a PC COM port through a null modem cable.

---

**NOTE:** The RS-232 settings in your program must match the settings specified in the front panel.System menu. Press the front panel <Shift> + <System> key if you need to change the settings.You can break data transmissions by sending a ^C or ^X character string to the multimeter. This clears any pending operation and discards any pending output.

---

### RS-232 Data Format

The RS-232 data is a 10-bit word with one start bit and one stop bit. The number of start and stop bits is not programmable. However, the following parity options are selectable using the front panel <Shift> + <System> key:

- **EVEN** Seven data bits with even parity
- **ODD** Seven data bits with odd parity
- **NONE** Eight data bits without parity

Parity options are stored in non-volatile memory.

### Baud Rate

The front panel <Shift> + <System> key lets you select one of the following baud rates, which is stored in non-volatile memory: 4800 9600 19200 38400 57600 115200

### RS-232 Flow Control

The RS-232 interface supports the following flow control options that are selected using the front panel <Shift> + <System> key. For each case, the electronic load will send a maximum of five characters after hold-off is asserted by the controller. The electronic load is capable of receiving as many as fifteen additional characters after it asserts hold-off.

**CTS/RTS** The electronic load asserts its Request to Send (RTS) line to signal hold-off when its input buffer is almost full, and it interprets its Clear to Send (CTS) line as a hold-off signal from the controller.

**XON/XOFF** When the input queue of the the electronic load becomes more than 3/4 full, the instrument issues an X\_OFF command. The control program should respond to this and stop sending characters until The electronic load issues the X\_ON, which it will do once its input buffer has dropped below half-full. The electronic load recognizes X\_ON and X\_OFF sent from the controller. An X\_OFF will cause the electronic load to stop outputting characters until it sees an X\_ON.

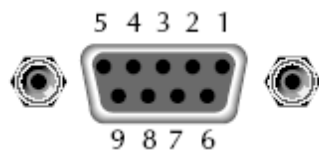
**NONE** There is no flow control.

Flow control options are stored in non-volatile memory.

## RS-232 connections

The RS-232 serial port can be connected to the serial port of a controller (i.e., personal computer) using a straight through RS-232 cable terminated with DB-9 connectors. **Do not use a null modem cable.** and table below shows the pinout for the connector.

If your computer uses a DB-25 connector for the RS-232 interface, you will need a cable or adapter with a DB-25 connector on one end and a DB-9 connector on the other, wired straight through (not null modem).



RS-232 connector pinout

pin number	description
1	no connection
2	TXD, transmit data
3	RXD, receive data
4	no connection
5	GND, signal ground
6	no connection
7	CTS, clear to send
8	RTS, ready to send
9	no connection

## RS-232 troubleshooting

If you are having trouble communicating over the RS-232 interface, check the following:

- The computer and the electronic load must be configured for the same baud rate, parity, number of data bits, and flow control options. Note that the electronic load is configured for 1 start bit and 1stop bit (these values are fixed).
- The correct interface cables or adapters must be used, as described under RS-232 connector. Note that even if the cable has the proper connectors for your system, the internal wiring may be incorrect.
- The interface cable must be connected to the correct serial port on your

computer (COM1, COM2.etc.).

### 1.1.3 USB-TMC Capabilities of the Electronic Load

All electronic load functions are programmable over the USB.

The USB488 interface capabilities of the electronic load are described follow:

- The interface is a 488.2 USB488 interface.
- The interface accepts REN\_CONTROL, GO\_TO\_LOCAL, and LOCAL\_LOCKOUT requests.
- The interface accepts the MsgID = TRIGGER USBTMC command message and forwards TRIGGER requests to the Function Layer.

The USB488 device capabilities of the electronic load are described follow:

- The device understands all mandatory SCPI commands.
- The device is SR1 capable
- The device is RL1 capable
- The device is DT1 capable

## 1.2 Programming the Status Registers

You can use status register programming to determine the operating condition of the electronic load at any time. For example, you may program the electronic load to generate an interrupt (assert SRQ) when an event such as a current protection occurs. When the interrupt occurs, your program can then act on the event in the appropriate fashion.

Table defines the status bits. Figure shows the status register structure of the electronic load. The Standard Event, Status Byte, and Service Request Enable registers and the Output Queue perform standard GPIB functions as defined in the *IEEE 488.2 Standard Digital Interface for Programmable Instrumentation*. The Operation Status and Questionable Status registers implement functions that are specific to the electronic load.

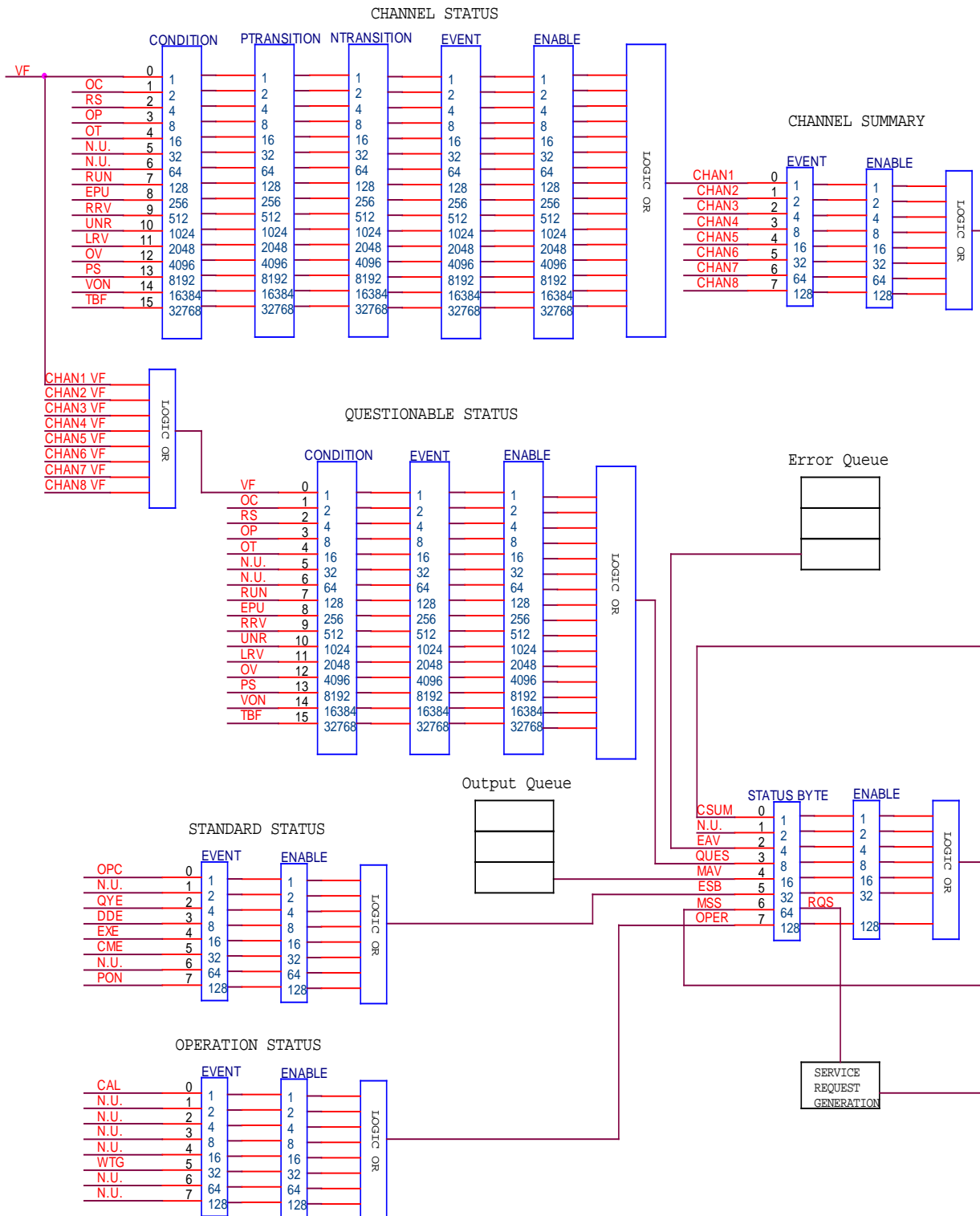
Bit	Signal	Meaning
0	<b>CAL</b>	<b>Operation Status Group</b> <i>Calibrating.</i> The electronic load is computing new calibration constants
5	<b>WTG</b>	<i>Waiting.</i> The electronic load is waiting for a trigger
0	<b>VF</b>	<b>Channel Status Group</b> <i>Voltage Fault.</i> Either an overvoltage or a reverse voltage has occurred. This bit reflects the active state of the FLT pin on the back of the unit. The bit remains set until the condition is removed and INP:PROT:CLE is programmed.
1	<b>OC</b>	<i>Over current.</i> An over-current condition has occurred. This occurs if the current exceeds 102% of the rated current or if it exceeds the user-programmed current protection level. Removing the over-current condition clears the bit. If the condition persists beyond the user programmable delay time, PS bit is also set and the input is



		turned off. Both bits remain set until the condition is removed and INP:PROT:CLE is programmed.
2	<b>RS</b>	<u>Remote Sense</u> when the real pannel sense is connected, this bit is true else false.
3	<b>OP</b>	<u>Overpower</u> . An overpower condition has occurred. This occurs if the unit exceeds the max power or it exceeds the user-programmed power protection level. Removing the overpower condition clears the bit. If the condition persists beyond the user programmable delay time, PS bit is also set and the input is turned off. Both bits remain set until the condition is removed and INP:PROT:CLE is programmed.
4	<b>OT</b>	<u>Over temperature</u> . An over-temperature condition has occurred. Both this bit and bit PS are set and the input is turned off. Both bits remain set until the unit is cooled down and INP:PROT:CLE is programmed.
7	<b>RUN</b>	<u>List run or stop status</u> when list is running, this bit is true else false.
8	<b>EPU</b>	<u>Extended Power Unavailable</u> . This bit is not used.
9	<b>RRV</b>	<u>Remote Reverse Voltage</u> . A reverse voltage condition has occurred on the sense terminals. Both this bit and VF bit are set. Removing the reverse voltage clears this bit but does not clear VF bit.VF Bit remains set until INP:PROT:CLE is programmed.
10	<b>UNR</b>	<u>Unregulated</u> . The input is unregulated. When the input is regulated the bit is cleared.
11	<b>LRV</b>	<u>Local Reverse Voltage</u> . A reverse voltage condition has occurred on the input terminals. Both this bit and VF bit are set. Removing the reverse voltage clears this bit but does not clear PS bit.PS Bit remains set until INP:PROT:CLE is programmed.
12	<b>OV</b>	<u>Over voltage</u> . An over voltage condition has occurred. Both this bit and VF bit are set and the load are turned off . Both bits remain set until the condition is removed and INP:PROT:CLE is programmed.
13	<b>PS</b>	<u>Protection Shutdown</u> . The protection shutdown circuit has tripped because of an Over-current, over-power, or over-temperature condition. The bit remains set until INP:PROT:CLE is programmed.
14	<b>VON</b>	<u>Voltage of sink current on</u> . When the voltage of input exceeds the user-programmed Von level, this bit is true else false.
15	<b>TBF</b>	<u>Trace Buffer Full</u> .
		<b>Questionable Status Group</b> <b>Same as Channel Status Group</b>
		<b>Standard Event Status Group</b>
0	<b>OPC</b>	<b>Operation Complete</b> . The load has completed all pending operations. *OPC must be programmed for this bit to be set when pending operations are complete.
2	<b>QYE</b>	<b>Query Error</b> . The output queue was read with no data present or the data was lost. Errors in the range of 499 through 400 can set this bit.
3	<b>DDE</b>	<b>Device-Dependent Error</b> . Memory was lost or self test failed. Errors in the range of 399 through 300 can set this bit.
4	<b>EXE</b>	<b>Execution Error</b> . A command parameter was outside its legal range, inconsistent with the load's operation, or prevented from executing because of an operating condition. Errors in the range of 299 through 200 can set this bit.
5	<b>CME</b>	<b>Command Error</b> . A syntax or semantic error has occurred or the load received a <get> within a program message. Errors in the range of 199 through 100 can set this bit.
7	<b>PON</b>	<b>Power-On</b> . The unit has been turned off and then on since this bit was last read.

<b>Status Byte and Service Request Enable Registers</b>		
0	<b>CSUM</b>	<b>Channel Summary. Indicates if an enabled channel event has occurred.</b>
2	<b>EAV</b>	<b>Error Available Summary. Indicates if the Error Queue contains data</b>
3	<b>QUES</b>	<b>Questionable Status Summary. Indicates if an enabled questionable event has occurred.</b>
4	<b>MAV</b>	<b>Message Available Summary. Indicates if the Output Queue contains data.</b>
5	<b>ESB</b>	<b>Event Status Summary. Indicates if an enabled standard event has occurred.</b>
6	<b>RQS/MSS</b>	<b>Master Status Summary. For an *STB? query, MSS is returned without being cleared.</b>
		<b>Request Service. During a serial poll, RQS is returned and cleared.</b>
7	<b>OPER</b>	<b>Operation Status Summary. Indicates if an operation event has occurred.</b>

## Load Status Register Structure



### 1.3 Condition registers

As Figure shows, all status register sets have a condition register. A condition register is a real-time, read-only register that constantly updates to reflect the current operating conditions of the instrument.

Use the :CONDition? query commands in the STATus Subsystem to read the condition registers.

## 1.4 Event registers

As Figure shows, each status register set has an event register. An event register is a latched, read-only register whose bits are set by the corresponding condition register, once a bit in an event register is set, it remains set (latched) until the register is cleared by a specific clearing operation. The bits of an event register are logically ANDed with the bits of the corresponding enable register and applied to an OR gate. The output of the OR gate is applied to the Status Byte Register.

Use the \*ESR? Common Command to read the Standard Event Register. All other event registers are read using the :EVENT? query commands in the STATus Subsystem.

An event register is cleared when it is read. The following operations clear all event registers:

- Cycling power
- Sending \*CLS

## 1.5 Enable registers

As Figure shows, each status register set has an enable register. An enable register is programmed by you and serves as a mask for the corresponding event register. An event bit is masked when the corresponding bit in the enable register is cleared (0). When masked, a set bit in an event register cannot set a bit in the Status Byte Register ( $1 \text{ AND } 0 = 0$ ).

To use the Status Byte Register to detect events (i.e., serial poll), you must unmask the events by setting (1) the appropriate bits of the enable registers. To program and query the Standard Event Status Register, use the \*ESE and \*ESE? Common Commands respectively. All other enable registers are programmed and queried using the :ENABLE and :ENABLE? commands in the STATus Subsystem.

An enable register is not cleared when it is read. The following operations affect the enable registers:

- Cycling power - Clears all enable registers
- :STATus:PREset clears the following enable registers:
  - ◆ Operation Event Enable Register
  - ◆ Questionable Event Enable Register
  - ◆ Channel Summary Event Enable Register
- \*ESE 0 - Clears the Standard Event Status Enable Register.

## 1.6 Queues

The Model 2000 uses two queues, which are first-in, first-out (FIFO) registers:

- Output Queue - used to hold reading and response messages
- Error Queue - used to hold error and status messages

The status model (Figure 4-5) shows how the two queues are structured with the other registers.

### 1.6.1 Output queue

The output queue holds data that pertains to the normal operation of the instrument. For example, when a query command is sent, the response message is placed on the Output Queue.

When data is placed in the Output Queue, the Message Available (MAV) bit in the Status Byte Register sets. A data message is cleared from the Output Queue when it is read. The Output Queue is considered cleared when it is empty. An empty Output Queue clears the MAV bit in the Status Byte Register.

Read a message from the Output Queue by addressing the electronic load to talk after the appropriate query is sent.

### 1.6.2 Error queue

The Error Queue holds error and status messages. When an error or status event occurs, a message that defines the error/status is placed in the Error Queue. This queue will hold up to 10 messages.

When a message is placed in the Error Queue, the Error Available (EAV) bit in the Status Byte Register is set. An error message is cleared from the Error/Status Queue when it is read. The Error Queue is considered cleared when it is empty. An empty Error Queue clears the EAV bit in the Status Byte Register. Read an error message from the Error Queue by sending either of the following SCPI query commands and then addressing the electronic load to talk:

```
:SYSTem:ERRor?.
```

## 1.7 Status Byte and Service Request (SRQ)

Service request is controlled by two 8-bit registers: the Status Byte Register and the Service Request Enable Register.

### 1.7.1 Status Byte Register

The summary messages from the status registers and queues are used to set or clear the appropriate bits (B0, B2, B3, B4, B5, and B7) of the Status Byte

Register. These bits do not latch, and their states (0 or 1) are solely dependent on the summary messages (0 or 1). For example, if the Standard Event Status Register is read, its register will clear. As a result, its summary message will reset to 0, which in turn will clear the ESB bit in the Status Byte Register.

Bit B6 in the Status Byte Register is either:

The Master Summary Status (MSS) bit, sent in response to the \*STB? Command, indicates the status of any set bits with corresponding enable bits set.

The Request for Service (RQS) bit, sent in response to a serial poll, indicates which device was requesting service by pulling on the SRQ line.

For a description of the other bits in the Status Byte Register, see “Common commands, \*STB?”

The IEEE-488.2 standard uses the following common query command to read the Status Byte

Register: \*STB?.

When reading the Status Byte Register using the \*STB? command, bit B6 is called the MSS bit. None of the bits in the Status Byte Register are cleared when using the \*STB? command to read it.

The IEEE-488.1 standard has a serial poll sequence that also reads the Status Byte Register and is better suited to detect a service request (SRQ). When using the serial poll, bit B6 is called the RQS bit. Serial polling causes bit B6 (RQS) to reset. Serial polling is discussed in more detail later in this section entitled “Serial Poll and SRQ.”

Any of the following operations clear all bits of the Status Byte Register:

- Cycling power.
- Sending the \*CLS common command

Note: The MAV bit may or may not be cleared.

## 1.7.2 Service request enable register

This register is programmed by you and serves as a mask for the Status Summary Message bits (B0, B2, B3, B4, B5, and B7) of the Status Byte Register. When masked, a set summary bit in the Status Byte Register cannot set bit B6 (MSS/RQS) of the Status Byte Register. Conversely, when unmasked, a set summary bit in the Status Byte Register sets bit B6.

A Status Summary Message bit in the Status Byte Register is masked when the corresponding bit in the Service Request Enable Register is cleared (0). When the masked summary bit in the Status Byte Register sets, it is ANDed with the corresponding cleared bit in the Service Request Enable Register. The logic “1” output of the AND gate is applied to the input of the OR gate and, thus, sets the MSS/RQS bit in the Status Byte Register.

The individual bits of the Service Request Enable Register can be set or cleared by using the following common command:

```
*SRE <NRf>
```

To read the Service Request Enable Register, use the \*SRE? query command. The Service Request Enable Register clears when power is cycled or a parameter (n) value of zero is sent with the \*SRE command \*SRE 0.

## 1.8 Serial poll and SRQ

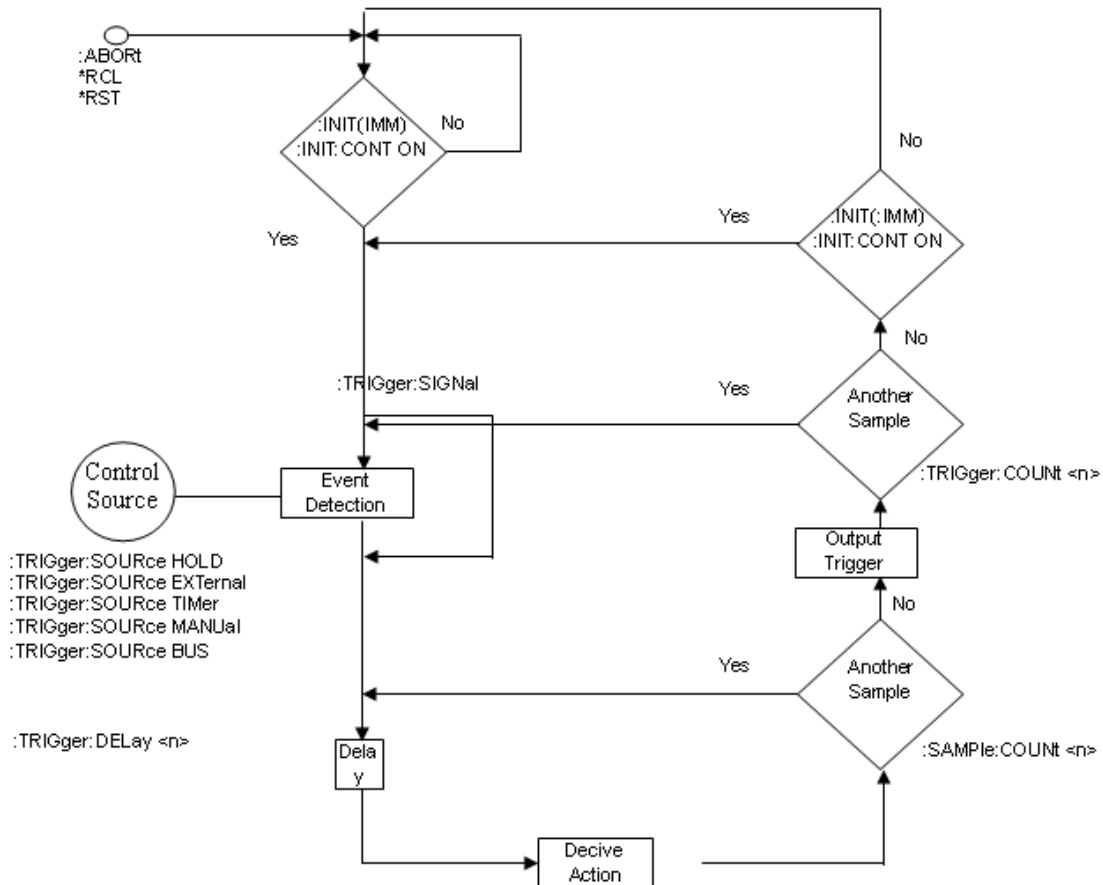
Any enabled event summary bit that goes from 0 to 1 will set RQS and generate a service request (SRQ). In your test program, you can periodically read the Status Byte Register to check if a service request (SRQ) has occurred and what caused it. If an SRQ occurs, the program can, for example, branch to an appropriate subroutine that will service the request. Typically, service requests (SRQs) are managed by the serial poll sequence of the electronic load. If an SRQ does not occur, bit B6 (RQS) of the Status Byte Register will remain cleared and the program will simply proceed normally after the serial poll is performed. If an SRQ does occur, bit B6 of the Status Byte Register will set and the program can branch to a service subroutine when the SRQ is detected by the serial poll.

The serial poll automatically resets RQS of the Status Byte Register. This allows subsequent serial polls to monitor bit B6 for an SRQ occurrence generated by other event types. After a serial poll, the same event can cause another SRQ, even if the event register that caused the first SRQ has not been cleared.

A serial poll clears RQS but does not clear MSS. The MSS bit stays set until all Status Byte event summary bits are cleared.

## 1.9 Trigger model (GPIB operation)

This section describes how the electronic load operates over the GPIB bus. The flowchart in figure below summarizes operation over the bus and is called the trigger model. It is called the trigger model because operation is controlled by SCPI commands from the Trigger subsystem. Key SCPI commands are included in the trigger model.



### 1.9.1 Idle and initiate

The instrument is considered to be in the idle state whenever it is not operating. While in the idle state, the instrument cannot perform any measure or scan functions. You can send two commands over the bus to remove the instrument from the idle state:

- :INITiate
- :INITiate:CONTInuous ON

With continuous initiation enabled (:INITiate:CONTInuous ON), the instrument will not remain in the idle state after all programmed operations are completed. However, you can return the instrument to the idle state at any time by sending any of these commands:

- \*RST
- ABORt
- \*RCL

### 1.9.2 Trigger model operation

Once the instrument is taken out of idle, operation proceeds through the trigger



model down to the device action.

**Control Source** — As shown in figure above, a control source is used to hold up operation until the programmed event occurs. The control source options are explained as follows:

- **HOLD** — Only the TRIG:IMM command will generate a trigger in HOLD mode. All other trigger commands are ignored.
- **MANual** — Event detection is satisfied by pressing the TRIG key.
- **TIMer** — This generates triggers that are in synchronization with the electronic load's internal oscillator as the trigger source. The internal oscillator begins running as soon as this command is executed. Use TRIG:TIM to program the oscillator period.
- **EXTernal** — Event detection is satisfied when an input trigger via the TRIGGER LINK connector is received by the electronic load.
- **BUS** — Event detection is satisfied when a bus trigger (GET or \*TRG) is received by the electronic load.
- **Delay** — A programmable delay is available after the event detection. The delay can be manually set from 0 to 999999.999 seconds.

## Chapter2 SCPI Introduction

### 2.1 SCPI Introduction

SCPI (Standard Commands for Programmable Instruments) is a programming language for controlling instrument functions over the GPIB and RS-232 and usb and ethernet interface. SCPI is layered on top of the hardware-portion of IEEE 488.2. The same SCPI commands and parameters control the same functions in different classes of instruments.

Conventions Used in This Guide:

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, NORM | TEXT indicates that either "TEXT" or "NORM" can be used as a parameter.

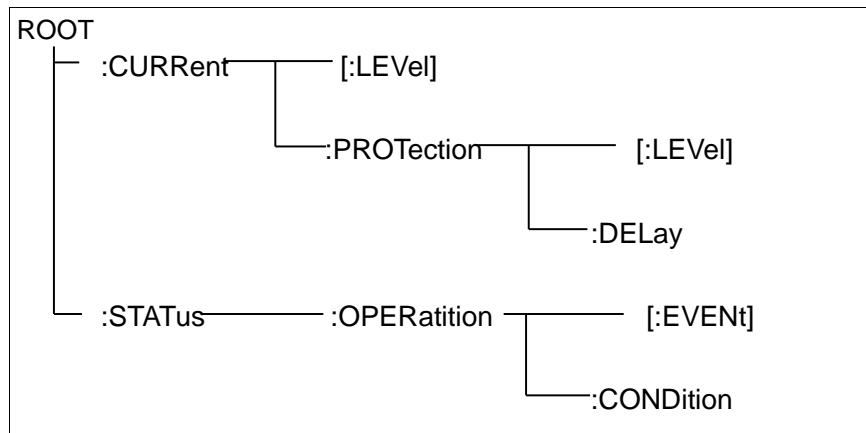
Square Brackets [ ] Items within square brackets are optional. The representation [SOURce:] VOLTage means that SOURce: may be omitted.

Braces { } Braces indicate parameters that may be repeated zero or more times. It is used especially for showing arrays. The notation <A>{<,B>} shows that parameter "A" must be entered, while parameter "B" may be omitted or may be entered one or more times.

### 2.2 Types of SCPI Commands

SCPI has two types of commands, common and subsystem.

- Common commands generally are not related to specific operation but to controlling overall electronic load functions, such as reset, status, and synchronization. All common commands consist of a three-letter mnemonic preceded by an asterisk: \*RST \*IDN? \*SRE 8
- Subsystem commands perform specific electronic load functions. They are organized into an inverted tree structure with the "root" at the top. The following figure shows a portion of a subsystem command tree, from which you access the commands located along the various paths.



Partial Command Tree

## 2.2.1 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 electronic load.

The header path can be thought of as a string that gets 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: `CURR:LEV 3;PROT:STAT OFF`, 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 "CURR" was omitted because after the "CURR:LEV 3" command, the header path became defined as "CURR" and thus the instrument interpreted the second command as: `CURR:PROT:STAT OFF`

In fact, it would have been syntactically incorrect to include the "CURR" explicitly in the second command, since the result after combining it with the header path would be: `CURR:CURR:PROT:STAT OFF`, which is incorrect.

## 2.2.2 Moving Among Subsystems

In order to combine commands from different subsystems, you need to be able to reset the header path to a null string within a message. You do this by beginning the command with a colon (:), which discards any previous header path. For example, you could clear the output protection and check the status of the Operation Condition register in one message by using a root specifier as

follows: PROTection:CLEAr;;STATus:OPERation:CONDition?

The following message shows how to combine commands from different subsystems as well as within the same subsystem: POWER:LEVel 200;PROTection 28; :CURRent:LEVel 3;PROTection:STATe ON

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

### 2.2.3 Including Common Commands

You can combine common commands with subsystem commands in the same 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.

VOLTage:TRIGgered 17.5;;INITialize;\*TRG

OUTPut OFF;\*RCL 2;OUTPut ON

### 2.2.4 Case sensitivity

Common commands and SCPI commands are not case sensitive. You can use upper or lower case and any case combination. Examples:

- \*RST = \*rst
- :DATA? = :data?
- :SYSTem:PRESet = :system:preset

### 2.2.5 Long-form and short-form versions

A SCPI command word can be sent in its long-form or short-form version. The command subsystem tables in Section 5 provide the in the long-form version. However, the short-form version is indicated by upper case characters. Examples:

- :SYSTem:PRESet long-form
- :SYST:PRES short form
- :SYSTem:PRES long-form and short-form combination

Note that each command word must be in long-form or short-form, and not something in between.

For example, :SYSTe:PRESe is illegal and will generate an error. The command will not be executed

## 2.2.6 Using Queries

Observe the following precautions with queries:

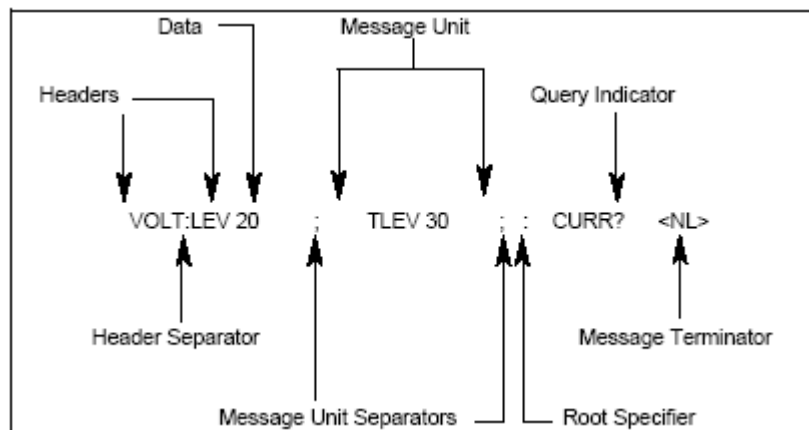
- Set up the proper number of variables for the returned data. For example, if you are reading back a measurement array, you must dimension the array according to the number of measurements that you have placed in the measurement buffer.
- Read back all the results of a query before sending another command to the electronic load. Otherwise a *Query Interrupted* error will occur and the unreturned data will be lost.

## 2.3 Types of 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 electronic load. The message, which may be sent at any time, requests the electronic load to perform some action.
- A response message consists of data in a specific SCPI format sent from the electronic load to the controller. The electronic load sends the message only when commanded by a program message called a "query."

The following figure illustrates SCPI message structure:



### 2.3.1 The Message Unit

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

ABORt<NL>

VOLTage 20<NL>

## 2.3.2 Headers

Headers, also referred to as keywords, are instructions recognized by the electronic load. Headers may be either in the long form or the short form. In the long form, the header is completely spelled out, such as VOLTAGE, STATUS, and DELAY. In the short form, the header has only the first three or four letters, such as VOLT, STAT, and DEL.

## 2.3.4 Query Indicator

Following a header with a question mark turns it into a query (VOLTage?, VOLTage:PROTection?). If a query contains a parameter, place the query indicator at the end of the last header(VOLTage:PROTection? MAX).

## 2.3.5 Message Unit Separator

When two or more message units are combined into a compound message, separate the units with a semicolon (STATus:OPERation?;QUEStionable?).

## 2.3.6 Root Specifier

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

## 2.3.7 Message Terminator

A terminator informs SCPI that it has reached the end of a message. Three permitted messages terminators are:

- newline (<NL>), which is ASCII decimal 10 or hex 0A.
- end or identify (<END>)
- both of the above (<NL><END>).

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

## 2.3.8 Command execution rules

- Commands execute in the order that they are presented in the program message.
- An invalid command generates an error and, of course, is not executed.

- Valid commands that precede an invalid command in a multiple command program message are executed.
- Valid commands that follow an invalid command in a multiple command program message are ignored.

## 2.4 SCPI Data Formats

All data programmed to or returned from the electronic load is ASCII. The data may be numerical or character string.

### Numerical Data Formats

Symbol	Data Form
<u>Talking Formats</u>	
<NR1>	Digits with an implied decimal point assumed at the right of the least-significant digit. Examples: <b>273</b>
<NR2>	Digits with an explicit decimal point. Example: <b>.0273</b>
<NR3>	Digits with an explicit decimal point and an exponent. Example: <b>2.73E+2</b>
<u>Listening Formats</u>	
<Nrf>	Extended format that includes <NR1>, <NR2> and <NR3>. Examples: <b>273 273. 2.73E2</b>
<Nrf+>	Expanded decimal format that includes <Nrf> and <b>MIN MAX DEF</b> . Examples: <b>273 273. 2.73E2 MAX. MIN</b> and <b>MAX</b> are the minimum and maximum limit values that are implicit in the range specification for the parameter. DEF is the default values for the parameter.
<Bool>	Boolean Data. Example: <b>0   1</b> or <b>ON   OFF</b>

### Suffixes and Multipliers

Class	Suffix	Unit	Unit with Multiplier
Amplitude	V	volt	MV (millivolt)
Current	A	ampere	MA (milliampere)
Power	W	watt	MW (milliwatt)
Resistance	ohm	OHM	MOHM (megohm)
	R	ohm	MR(megohm)
Slew Rate	A/uS	amps/second	-
	R/s	ohms/second	-
	V/s	volts/second	-
Time	s	second	MS (millisecond)

### Common Multipliers

1E3	K	kilo
1E-3	M	milli
1E-6	U	micro

## 2.5 Response Data Types

Character strings returned by query statements may take either of the following forms, depending on the length of the returned string:

**<CRD>** Character Response Data. Permits the return of character strings.

**<AARD>** Arbitrary ASCII Response Data. Permits the return of un delimited 7-bit ASCII. This data type has an implied message terminator.

**<SRD>** String Response Data. Returns string parameters enclosed in double quotes.

### Response messages

A response message is the message sent by the instrument to the computer in response to a query command program message.

### Sending a response message

After sending a query command, the response message is placed in the Output Queue. When the Model 2000 Multimeter is then addressed to talk, the response message is sent from the Output Queue to the computer.

### Multiple response messages

If you send more than one query command in the same program message (see the paragraph entitled, "Multiple Command Messages"), the multiple response messages for all the queries is sent to the computer when the Model 2000 is addressed to talk. The responses are sent in the order that the query commands were sent and are separated by semicolons (;). Items within the same query are separated by commas (,). The following example shows the response message for a program message that contains four single item query commands: 0; 1; 1; 0

### Response message terminator (RMT)

Each response is terminated with an LF (line feed) and EOI (end or identify). The following example shows how a multiple response message is terminated: 0; 1; 1; 0; <RMT>



## Message exchange protocol

Two rules summarize the message exchange protocol:

- **Rule 1.** You must always tell the Model 2000 what to send to the computer.

The following two steps must always be performed to send information from the instrument other computer:

1. Send the appropriate query command(s) in a program message.
2. Address the Model 2000 to talk.

- **Rule 2.** The complete response message must be received by the computer before another program message can be sent to the Model 2000.

## 2.6 SCPI Command Completion

SCPI commands sent to the electronic load are processed either sequentially or in parallel. Sequential commands finish execution before a subsequent command begins. Parallel commands allow other commands to begin executing while the parallel command is still executing. Commands that affect trigger actions are among the parallel commands.

The \*WAI, \*OPC, and \*OPC? common commands provide different ways of indicating when all transmitted commands, including any parallel ones, have completed their operations. The syntax and parameters for these commands are described in chapter 4. Some practical considerations for using these commands are as follows:

\*WAI-This prevents the electronic load from processing subsequent commands until all pending operations are completed.

\*OPC?-This places a 1 in the Output Queue when all pending operations have completed. Because 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-This 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 system must be in the Idle state in order for the status OPC bit to be true. Therefore, as far as triggers are concerned, OPC is false whenever the trigger system is in the Initiated state.

---

## Using Device Clear

You can send a device clear at any time to abort a SCPI command that may be hanging up the GPIB interface. The status registers, the error queue, and all configuration states are left unchanged when a device clear message is received. Device clear performs the following actions:

- ◆□ The input and output buffers of the electronic load are cleared.
- ◆□ The electronic load is prepared to accept a new command string.

The following statement shows how to send a device clear over the GPIB interface using *GW BASIC*:

```
CLEAR 705                IEEE-488 Device Clear
```

The following statement shows how to send a device clear over the GPIB interface using the GPIB command library for *C* or *QuickBASIC*:

```
IOCLEAR (705)
```

## 2.7 SCPI Conformance Information

### 2.7.1 Language Dictionary Introduction

This section gives the syntax and parameters for all the IEEE 488.2 SCPI subsystem and common commands used by the electronic loads. Because the SCPI syntax remains the same for all programming languages, the examples given for each command are generic.

<b>Syntax Forms</b>	Syntax definitions use the long form, but only short form headers (or "keywords") appear in the examples. Use the long form to help make your program selfdocumenting.
<b>Parameters</b>	Most commands require a parameter and all queries will return a parameter. The range for a parameter may vary according to the model of electronic load. Parameters for all models are listed in the Specifications table in the User's Guide.
<b>Channel</b>	If a command only applies to individual channels of a mainframe, the entry Channel Selectable will appear in the command description.
<b>Related Commands</b>	Where appropriate, related commands or queries are included. These are listed because they are either directly related by function, or because reading about them will clarify or enhance your understanding of the original command or query.
<b>Order of Presentation</b>	The dictionary is organized as follows: Subsystem commands, arranged by subsystem_ IEEE 488.2 common commands

## 2.8 Subsystem Commands

Subsystem commands are specific to 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. The description of common commands follows the description of the subsystem commands.

The subsystem command groups are arranged according to function: Calibration, Channel, Input, List, Measurement, Port, Status, System, Transient, and Trigger. Commands under each function are grouped alphabetically under the subsystem. Commands followed by a question mark (?) take only the query form.

When commands take both the command and query form, this is noted in the syntax descriptions.

## Chapter3 Programming Examples

This chapter displays the programming examples to remotely control IT8700 load using SCPI commands.

### Note

- ◆ If the user want to change the settings of the instrument, for instance, the input setting value, the command SYST:REM must be sent to the instrument after finishing the connection between the instrument and PC.
- ◆ “ - >” indicates the commands that you send to the IT8700 load supply.

### Example 1: Identifying the Load in Use

You can verify whether you are communicating with the right IT8700 load.

To query the identification of the load, send the command:

```
-> *IDN?
```

To check the power supply error queue, send the command:

```
-> SYST:ERR?
```

### Example 2: Common input commands

```
-> SYSTem:REMOte
```

```
-> FUNCTion CURRent
```

```
-> CURRent 3
```

```
-> FUNCTion VOLTage
```

```
-> VOLTage 10
```

```
-> FUNCTion POWer
```

```
-> POWer 10
```

```
-> FUNCTion RESistance
```

```
-> RESistance:LED 1
```

```
-> RESistance:VDRop 10
```

```
-> RESistance 10
```

```
-> INPut ON
```

```
-> MEASure:VOLTage?
```

```
-> MEASure:CURRent?
```

-> MEASure:POWer?

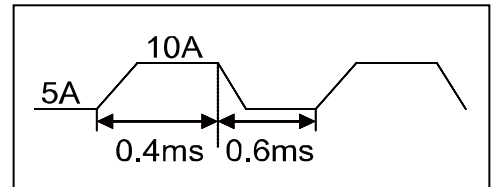
## Example 3: Programming Transients

Transient operation is used to synchronize input changes with internal or external trigger signals, and simulate loading conditions with precise control of timing, duration, and slew. The following transient modes can be generated:

### Continuous Transients

In continuous operation, a repetitive pulse train switches between two load levels. The rate at which the level changes is determined by the slew rate (see slew rate descriptions for CV, CR, or CV mode as applicable). In addition, Use the following commands to program continuous transients:

```
CURRent:TRANSient:MODE CONTInuous
CURRent:TRANSient:ALEVel 5
CURRent:TRANSient:AWIDth 0.6mS
CURRent:TRANSient:BLEVel 10
CURRent:TRANSient:BWIDth 0.4mS
TRANSient ON
INPut ON
TRIGGer:IMMediate
```

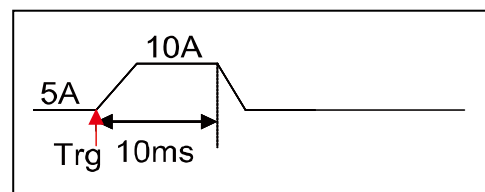


### Pulse Transients

Pulsed transient operation generates a load change that returns to level B state after some time period.

Use the following commands to program pulsed transients:

```
CURRent:TRANSient:MODE PULSE
CURRent:TRANSient:ALEVel 10
CURRent:TRANSient:BLEVel 5
CURRent:TRANSient:AWIDth 10mS
TRANSient ON
INPut ON
TRIGGer:IMMediate
```

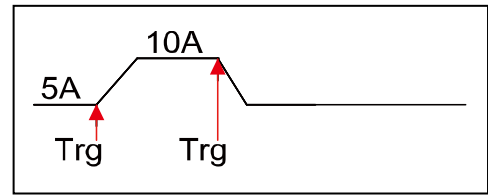


### Toggled Transients

Under toggle mode, after enabling dynamic test operation, the load will be switched continuously between A value and B value after receipt of every trigger signal. Use the following commands to program toggled transients:

```

CURRent:TRANsient:MODE TOGGLE
CURRent:TRANsient:ALeVel 10
CURRent:TRANsient:BLeVel 5
TRANsient ON
INPut ON
TRIGger:IMMediate
    
```



## Example 4: Programming Lists

List mode lets you generate complex sequences of input changes with rapid, precise timing, which may be synchronized with internal or external signals. This is useful when running test sequences with a minimum amount of programming overhead.

You can program up to 84 settings (or steps) in the list, the time interval (dwell) that each setting is maintained, the number of times that the list will be executed, and how the settings change in response to triggers. All list data is can be stored in nonvolatile memory using the LIST:SAV command. This means that the programmed data for any list will be retained when the electronic load is turned off. Use the LIST:RCL command to recall the saved state.

The following procedure shows how to generate a simple 4-step list of current changes.

```

FUNC CURRent
LIST:RANGe 40
LIST:COUNT 10000
LIST:STEP 4
LIST:LEVel 1, 5
LIST:SLEW 1, 1
LIST:WIDth 1, 10ms
LIST:LEVel 2, 10
LIST:SLEW 2, 1
LIST:WIDth 2, 10ms
LIST:LEVel 3, 20
LIST:SLEW 3, 1
LIST:WIDth 3, 10ms
LIST:LEVel 4, 15
    
```

```
LIST:SLEW 4, 1  
LIST:WIDTH 4, 10ms  
FUNCTION:MODE LIST  
TRIGGER:IMMEDIATE
```

## Example 5: Trace function

The following is an example of how to use the commands of the Trace subsystem:

```
TRACE:CLEAR //Clear the buffer of readings.  
TRACE:POINTS 2000 //Specify the size of the buffer.  
TRACE:FEED TWO //Select the source of readings to be placed in the buffer.  
TRACE:FEED:CONTROL NEXT //Select the buffer control.  
TRACE:TIMER 0.00002 //Select the interval for timer.  
TRACE:DELAY 1 //Select the delay time for trigger in buffer.  
TRIGGER //Trigger the instrument to enter data storage status.  
TRACE:DATA? //Read the data stored in the buffer to the PC interface.
```

## Chapter4 IEEE-488 Command Reference

Common commands begin with an \* and consist of three letters (command) or three letters and a ?(query). They are defined by the IEEE 488.2 standard to perform common interface functions. Common commands and queries are categorized under System, Status, or Trigger functions and are listed at the end of this chapter.

### Common Commands

Common commands begin with an \* and consist of three letters (command) IEEE 488.2 standard to perform some common interface functions. The electronic loads respond to the required common commands that control status reporting, synchronization, and internal operations. The electronic loads also respond to optional common commands that control triggers, power-on conditions, and stored operating parameters.

Common commands and queries are listed alphabetically. If a command has a corresponding query that simply returns the data or status specified by the command, then both command and query are included under the explanation for the command. If a query does not have a corresponding command or is functionally different from the command, then the query is listed separately. The description for each common command or query specifies any status registers affected. Refer to chapter “Programming the Status Registers”, which explains how to read specific register bits and use the information that they return.

Mnemonic	Meaning	Description
*CLS	Clear status	Clears all event registers and Error Queue.
*ESE <NRf>	Event enable command	Program the Standard Event Enable Register.
*ESE?	Event enable query	Read the Standard Event Enable Register.
*ESR?	Event status query	Read the Standard Event Status Register and clear it.
*IDN?	Identification query	Return the manufacturer, model number, serial number and firmware revision levels of the unit.
*RDT?	Frame query	Return the type of electronic frame.
*OPC	Operation complete command	Set the Operation Complete bit in the Standard Event Status Register after all pending commands have been executed.
*OPC?	Operation complete query	Places an ASCII “1” into the output queue when all pending selected device operations have been completed.
*RCL <NRf>	Recall Command	Returns the Load to the setup configuration stored in the specified memory location.
*RST	Reset Command	Returned the Load to the *RST default conditions.



*SAV <NRf>	Save Command	Saves the current setup to the specified memory location.
*SRE <NRf>	Service request enable command	Programs the Service Request Enable register.
*SRE?	Service request enable query	Reads the Service Request Enable Register.
*STB?	Read status byte query	Read the Status Byte Register.
*TRG	Trigger Command	Send a trigger to Load.
*TST?	Self-test query	Performs a self-test and returns the result.
*WAI	Wait to continue command	Wait until all previous commands are executed.

## \*CLS — Clear Status

This command clears the bits of the following registers:

- \_ Standard Event Register
- \_ Operation Event Register
- \_ Questionable Event Register
- \_ Channel Summary Event Register
- \_ Channel Event Register
- \_ Error Queue

### Command Syntax

\*CLS

### Parameters

None

## \*ESE <NRf> — Event Enable

This command programs the Standard Event Status Enable register bits. The programming determines which events of the Standard Event Status Event register (see \*ESR?) are allowed to set the ESB (Event Summary Bit) of the Status Byte register. A "1" in the bit position enables the corresponding event. All of the enabled events of the Standard Event Status Event Register are logically ORed to cause the Event Summary Bit (ESB) of the Status Byte Register to be set. See chapter "Programming the Status Registers" for descriptions of the Standard Event Status registers.

The query reads the Standard Event Status Enable register.

## Command Syntax

\*ESE <NRf>

## Parameters

0 to 255

## Power-On Value

see \*PSC

## Examples

\*ESE 129

## Query Syntax

\*ESE?

## Returned Parameters

<NR1>

## Related Commands

\*ESR? \*PSC \*STB?

## **\*ESR?**

This query reads the Standard Event Status Event register. Reading the register clears it. The bit configuration of this register is the same as the Standard Event Status Enable register (see \*ESE). See chapter “Programming the Status Registers” for a detailed explanation of this register.

## Query Syntax

\*ESR?

## Parameters

None

## Returned Parameters

<NR1> (register value)

## Related Commands

\*CLS \*ESE \*ESE? \*OPC

## \*IDN?

This query requests the electronic load to identify itself. It returns the data in four fields separated by commas.

### Query Syntax

\*IDN?

### Parameters

None

### Returned Parameters

<AARD>	Field	Information
	ITECH	Technologies manufacturer
	xxxx	model number
	nnnn	serial number or 0
	x.xx	firmware revision

### Example

ITECH Ltd., IT8700, 002031, 1.01

## \*RDT?

This query requests the types of Electronic Ferame. If channel does not exist, it returns 0. If channel exist, it returns the types like IT8722P.

### Query Syntax

\*RDT?

### Parameters

None

### Returned Parameters

<AARD>

### Example

IT8722P, IT8722P, 0, 0, 0, 0, 0, 0

## \*OPC

This command causes the interface to set the OPC bit (bit 0) of the Standard Event Status register when the electronic load has completed all pending operations. (See \*ESE for the bit configuration of the Standard Event Status

registers.) Pending operations are complete when:

- All commands sent before \*OPC have been executed. This includes overlapped commands. Most commands are sequential and are completed before the next command is executed. Overlapped commands are executed in parallel with other commands. Commands that affect trigger actions are overlapped with subsequent commands sent to the electronic load. The \*OPC command provides notification that all overlapped commands have been completed.
- All triggered actions are completed and the trigger system returns to the Idle state.

\*OPC does not prevent processing of subsequent commands but Bit 0 will not be set until all pending operations are completed. The query causes the interface to place an ASCII "1" in the Output Queue when all pending operations are completed.

### Command Syntax

\*OPC

### Parameters

None

### Query Syntax

\*OPC?

### Returned Parameters

<NR1>

### Related Commands

\*TRG

\*WAI

## \*PSC

This command is used to control whether the electronic load will generate a service request when power on again.

1 OR ON: When the load power on, status byte enable register, operator event enable register, query event enable register and standard event enable register will be cleared.

0 OR OFF: The value of status byte enable register, operator event enable register, query event enable register and standard event enable register will be stored in the non-volatile storage, which will be recalled when power on.

### Command Syntax

\*PSC <bool>

### Parameters

0|1|ON|OFF

### Query Syntax

\*PSC?

### Returned Parameters

0|1

## \*RCL

This command restores the electronic load to a state that was previously stored in memory with a \*SAV command to the specified location. All states are recalled with the following exceptions:

CAL:STATE is set to OFF

The trigger system is set to the Idle state by an implied ABORt command (this cancels any uncompleted trigger actions)

**NOTE:** The device state stored in location 0 is automatically recalled at power turn-on.

### Command Syntax

\*RCL <NRf>

### Parameters

0 to 100

### Example

\*RCL 3

### Related Commands

\*PSC

\*RST

\*SAV

## \*RST

This command resets **ALL** channels of the electronic load to the following

factory-defined states:

INP OFF	INP:SHOR OFF
TRAN OFF	FUNC:MODE FIX
FUNC CURR	CURR:RANG MAX
CURR MIN	CURR:SLEW MAX
CURR:TRAN:ALEV MAX	CURR:TRAN:AWID MIN
CURR:TRAN:BLEV MIN	CURR:TRAN:BWID MIN
CURR:TRAN:MODE CONT	VOLT:RANG MAX
VOLT MAX	
VOLT:TRAN:ALEV MAX	VOLT:TRAN:AWID MIN
VOLT:TRAN:BLEV MIN	VOLT:TRAN:BWID MIN
VOLT:TRAN:MODE CONT	RES:RANG MAX
RES MAX	
RES:TRAN:ALEV MAX	RES:TRAN:AWID MIN
RES:TRAN:BLEV MIN	RES:TRAN:BWID MIN
RES:TRAN:MODE CONT	CURR:PROT:STAT OFF
CURR:PROT:LEV MAX	CURR:PROT:DEL 3
POW:CONF MAX	POW:PROT:LEV MAX
POW:PORT:DEL 3	VOLT:LATC OFF
VOLT:ON MIN	SENS:VOLT:RANG:AUTO ON
SENS:AVER:COUN 14	
TRAC:POIN 0	TRAC:FEED TWO
TRAC:FEED:CONT NEV	

**NOTE:** \*RST does not clear any of the status registers or the error queue, and does not affect any interface error conditions.

\*RST sets the trigger system to the Idle state.

\*RST clears the presently active list.

## Command Syntax

\*RST

## Parameters

None

## Related Commands

\*PSC

\*SAV

## \*SAV

This command stores the present state of the electronic load to a specified location in memory. Up to 100 states can be stored.

If a particular state is desired at power-on, it should be stored in location 0. It then will be recalled at power-on if the power-on state is set to RCL0. Use \*RCL to retrieve instrument states.

NOTE: \*SAV does not save the programmed trigger values ([SOURce:]CURRent:TRIGGer,[SOURce:]RESistance:TRIGGer,[SOURce:]VOLtage:TRIGGer). Programming an \*RCL or a \*RST command causes the triggered settings to revert to their [IMMEDIATE] settings.

## Command Syntax

\*SAV <NRf>

## Parameters

0 to 100

## Example

\*SAV 3

## Related Commands

\*PSC

\*RST

\*RCL

## \*SRE

This command sets the condition of the Service Request Enable Register. This register determines which bits from the Status Byte Register (see \*STB for its bit configuration) are allowed to set the Master Status Summary (MSS) bit and the Request for Service (RQS) summary bit. A 1 in any Service Request Enable Register bit position enables the corresponding Status Byte Register bit and all such enabled bits then are logically ORed to cause 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 electronic load cannot generate an SRQ to the controller. The query returns the current state of \*SRE.

### Command Syntax

\*SRE <NRf>

### Parameters

0 to 255

### Default Value

see \*PSC

### Example

\*SRE 128

### Query Syntax

\*SRE?

### Returned Parameters

<NR1> (register binary value)

### Related Commands

\*ESE

\*ESR

\*PSC

## **\*STB?**

This query reads the Status Byte register, which contains the status summary bits and the Output Queue MAV bit. Reading the Status Byte register does not clear it. The input summary bits are cleared when the appropriate event registers are read (see chapter “Programming the Status Registers” for more information). 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 electronic load has one or more reasons for requesting service.

### Query Syntax

\*STB?



### Parameters

None

### Returned Parameters

<NR1> (register value)

### Related Commands

\*SRE

\*ESR

\*ESE

## **\*TRG**

This command generates a trigger to any system that has BUS selected as its source (for example, TRIG:SOUR BUS). The command has the same affect as the Group Execute Trigger (<GET>) command.

### Command Syntax

\*TRG

### Parameters

None

### Related Commands

TRIG:IMM

## **\*TST?**

This query causes the electronic load to do a self-test and report any errors.

### Query Syntax

\*TST?

### Parameters

None

### Returned Parameters

<NR1> 0 indicates the electronic load has passed selftest.

Non-zero indicates an error code (see appendix C)

## **\*WAI**

This command instructs the electronic load not to process any further commands until all pending operations are completed. Pending operations are complete when:

All commands sent before \*WAI have been executed. This includes overlapped commands. Most commands are sequential and are completed before the next command is executed. Overlapped commands are executed in parallel with other commands. Commands that affect input voltage or state, relays, and trigger actions are overlapped with subsequent commands sent to the electronic load. The \*WAI command prevents subsequent commands from being executed before any overlapped commands have been completed.

All triggered actions are completed and the trigger system returns to the Idle state. \*WAI can be aborted only by sending the electronic load a GPIB DCL (Device Clear) command.

### Command Syntax

\*WAI

### Parameters

None

### Related Commands

\*OPC

---

## Chapter5 System Commands

---

System commands control the system-level functions of the electronic load that are not directly related to input control or measurement functions.

### **SYSTem:PRESet**

This command returns the instrument to states optimized for front panel operation.

#### Command Syntax

SYSTem:PRESet

#### Parameters

None

### **SYSTem:POSetup**

This command is used to select the power-on defaults. With RST selected, the instrument powers up to the \*RST default conditions. With the SAV0 parameter selected, the instrument powers-on to the setup that is saved in the specified location using the \*SAV command.

#### Command Syntax

SYSTem:POSetup <CRD>

#### Parameters

RST | SAV0

#### \*RST Value

RST

#### Examples

SYST:POS RST

#### Query Syntax

SYSTem:POSetup?

#### Returned Parameters

<CRD>

## Related Commands

\*RST \*SAV

## SYSTem:VERSion?

This query returns the SCPI version number to which the electronic load complies. The value is of the form YYYY.V, where YYYY is the year and V is the revision number for that year.

### Query Syntax

SYSTem:VERSion?

### Parameters

None

### Examples

SYST:VERS?

### Returned Parameters

<NR2>

## SYSTem:ERRor?

This query returns the next error number followed by its corresponding error message string from the remote programming error queue. The queue is a FIFO (first-in, first-out) buffer that stores errors as they occur. As it is read, each error is removed from the queue. When all errors have been read, the query returns "0, No Error". If more errors are accumulated than the queue can hold, the last error in the queue is "-350, Too Many Errors".

### Query Syntax

SYSTem:ERRor?

### Parameters

None

### Examples

SYST:ERR?

### Returned Parameters

<NR1>, <SRD>

## **SYSTem:CLEar**

This action command is used to clear the Error Queue of messages.

### Command Syntax

SYSTem:CLEar

### Parameters

None

### Examples

SYST:CLE

### Related Commands

SYST:ERR?

## **SYSTem:LOCal**

This command places the electronic load in local mode during remote operation. The front panel keys are functional.

### Command Syntax

SYSTem:LOCal

### Parameters

None

### Examples

SYST:LOC

### Related Commands

SYST:REM    SYST:RWL

## **SYSTem:REMote**

This command places the electronic load in remote mode during remote operation. This disables all front panel keys except the Local key. Pressing the Local key while in the remote state returns the front panel to the local state.

### Command Syntax

SYSTem:REMote

**Parameters**

None

**Examples**

SYST:REM

**Related Commands**

SYST:LOC    SYST:RWL

**SYSTem:RWLock**

This command places the electronic load in remote mode during remote operation. All front panel keys including the Local key are disabled. Use SYSTem:LOCal to return the front panel to the local state.

**Command Syntax**

SYSTem:RWLock

**Parameters**

None

**Examples**

SYST:RWL

**Related Commands**

SYST:REM    SYST:LOC

---

## Chapter6 STATus Subsystem

---

These commands program the electronic load status registers. The electronic load has five groups of status registers; Channel Status, Channel Summary, Questionable Status, Standard Event Status, and Operation Status. Refer to “Programming the Status Registers” for more information.

### **STATus:CHANnel?**

Channel Specific

This query returns the value of the Channel Event register. The Event register is a read-only register which holds (latches) all events that are passed into it. Reading the Channel Event register clears it.

#### Query Syntax

STATus:CHANnel[:EVENT]?

#### Parameters

None

#### Examples

STAT:CHAN:EVEN?

#### Returned Parameters

<NR1> (register value)

#### Related Commands

\*CLS

### **STATus:CHANnel:CONDition?**

Channel Specific

This query returns the value of the Channel Condition register. The particular channel must first be selected by the CHAN command.

#### Query Syntax

STATus:CHANnel:CONDition?

#### Parameters

None

### Examples

STAT:CHAN:COND?

### Returned Parameters

<NR1> (register value)

### Related Commands

STAT:CHAN?

## **STATus:CHANnel:ENABLE**

Channel Specific

This command sets or reads the value of the Channel Enable register for a specific channel. The particular channel must first be selected by the CHAN command.

### Command Syntax

STATus:CHANnel:ENABLE <NR1>

### Parameters

0 to 65535

### Examples

STAT:CHAN:ENAB 3

### Query Syntax

STATus:CHANnel:ENABLE?

### Returned Parameters

<NR1> (register value)

### Related Commands

\*CLS

## **STATus:CSUMary:EVENT?**

This query returns the value of the Channel Event summary register. The bits in this register correspond to a summary of the channel register for each input channel. Reading the Channel Event summary register clears it. This command is not channel specific, it applies to the entire mainframe.



### Query Syntax

STATus:CSUMary:EVENT?

### Parameters

None

### Examples

STAT:CSUM:EVENT?

### Returned Parameters

<NR1> (register value)

### Related Commands

\*CLS

## **STATus:CSUMmary:ENABLE**

This command sets or reads the value of the Channel Enable summary register. This command is not channel specific, it applies to the entire mainframe.

### Command Syntax

STATus:CSUMmary:ENABLE <NR1>

### Parameters

0 to 255

### Examples

STAT:CSUM:ENAB 3

### Returned Parameters

<NR1> (register value)

### Related Commands

\*CLS

## **STATus:OPERation?**

This query returns the value of the Operation Event register. The Event register is a read-only register that holds (latches) all events that are passed by the Operation NTR and/or PTR filter. Reading the Operation Event register clears it. This command is not channel specific, it applies to the entire mainframe.

**Query Syntax**

STATus:OPERation[:EVENT]?

**Parameters**

None

**Examples**

STAT:OPER:EVEN?

**Returned Parameters**

&lt;NR1&gt; (register value)

**Related Commands**

\*CLS

## **STATus:OPERation:CONDition?**

This query returns the value of the Operation Condition register. That is a read-only register that holds the real-time (unlatched) operational status of the electronic load. This command is not channel specific, it applies to the entire mainframe.

**Query Syntax**

STATus:OPERation:CONDition?

**Parameters**

None

**Examples**

STAT:OPER:COND?

**Returned Parameters**

&lt;NR1&gt; (register value)

**Related Commands**

STAT:QUES:COND?

## **STATus:OPERation:ENABLE**

This command and its query set and read the value of the Operation Enable register. This register is a mask for enabling specific bits from the Operation Event register to set the operation summary bit (OPER) of the Status Byte register. The operation summary bit is the logical OR of all enabled Operation

Event register bits. This command is not channel specific, it applies to the entire mainframe.

### Command Syntax

STATus:OPERation:ENABLE <NR1>

### Parameters

0 to 65535

### Default Value

0

### Examples

STAT:OPER:ENAB 32 STAT:OPER:ENAB 1

### Query Syntax

STATus:OPERation:ENABLE?

### Returned Parameters

<NR1> (register value)

### Related Commands

STAT:OPER?

## **STATus:QUEStionable?**

This query returns the value of the Questionable Event register. The Event register is a read-only register that holds (latches) all events that pass into it. Reading the Questionable Event register clears it. This command is not channel specific, it applies to the entire mainframe.

### Query Syntax

STATus:QUEStionable[:EVENT]?

### Parameters

None

### Examples

STAT:QUES:EVEN?

### Returned Parameters

<NR1> (register value)

## Related Commands

\*CLS

## STATus:QUEStionable:CONDition?

This query returns the value of the Questionable Condition register. That is a read-only register that holds the real-time (unlatched) questionable status of the electronic load. This command is not channel specific, it applies to the entire mainframe.

### Query Syntax

STATus:QUEStionable:CONDition?

### Parameters

None

### Examples

STAT:QUES:COND?

### Returned Parameters

<NR1> (register value)

## Related Commands

STAT:OPER:COND?

## STATus:QUEStionable:ENABLE

This command sets or reads the value of the Questionable Enable register. This register is a mask for enabling specific bits from the Questionable Event register to set the questionable summary (QUES) bit of the Status Byte register. This bit (bit 3) is the logical OR of all the Questionable Event register bits that are enabled by the Questionable Status Enable register. This command is not channel specific, it applies to the entire mainframe.

### Command Syntax

STATus:QUEStionable:ENABLE <NR1>

### Parameters

0 to 65535

### Default Value

0

## Examples

STAT:QUES:ENAB 32 STAT:QUES:ENAB 1

## Query Syntax

STATus:QUEStionable:ENABLE?

## Returned Parameters

<NR1> (register value)

## Related Commands

STAT:QUES?

# STATus:PRESet

When this command is sent, the SCPI event registers are affected as follows:

All bits of the following registers are cleared to zero (0):

- Questionable Event Enable Register.
- Channel summary Event Enable Register.
- Operation Event Enable Register

NOTE: Registers not included in the above list are not affected by this command.

## Command Syntax

STATus:PRESet

## Parameters

None

## Examples

STAT:PRES

---

## Chapter7 SCPI Measurement Command

---

### **FETCh:VOLTage[:DC]?**

This command is used to get the latest measured DC voltage.

### **MEASure:VOLTage[:DC]?**

This command initiates and executes a new voltage measurement, and returns the measured input voltage of the electronic load.

#### Command Syntax

FETCh:VOLTage[:DC]?

MEASure:VOLTage[:DC]?

#### Parameters

None

#### Examples

FETC:VOLT?

#### Returned Parameters

<NRf>

### **FETCh:VOLTage:MAX?**

This command is used to read the max voltage of the latest measurement.

### **MEASure:VOLTage:MAX?**

This command initiates and executes a new maximum voltage measurement, and returns the measured input maximum voltage of the electronic load.

#### Command Syntax

FETCh:VOLTage:MAX?

MEASure:VOLTage:MAX?

#### Parameters

None

## Returned Parameters

<NRf>

## **FETCh:VOLTage:MIN?**

This command is used to read the min voltage of the latest measurement.

## **MEASure:VOLTage:MIN?**

This command initiates and executes a new minimum voltage measurement, and returns the measured input minimum voltage of the electronic load.

## Command Syntax

FETCh:VOLTage:MIN?

MEASure:VOLTage:MIN?

## Parameters

None

## Returned Parameters

<NRf>

## **FETCh:CURRent[:DC]?**

This command is used to get the latest measured DC current.

## **MEASure:CURRent[:DC]?**

This command initiates and executes a new current measurement, and returns the measured input current of the electronic load.

## Command Syntax

FETCh:CURRent[:DC]?

MEASure:CURRent[:DC]?

## Parameters

None

## Examples

FETC:CURR?

MEAS:CURR?

## Returned Parameters

<NRf>

## **FETCh:CURRent:MAX?**

This command is used to read the max voltage of the latest measurement.

## **MEASure:CURRent:MAX?**

This command initiates and executes a new maximum current measurement, and returns the measured input maximum current of the electronic load.

## Command Syntax

FETCh:CURRent:MAX?

MEASure:CURRent:MAX?

## Parameters

None

## Returned Parameters

<NRf>

## **FETCh:CURRent:MIN?**

This command is used to read the min voltage of the latest measurement.

## **MEASure:CURRent:MIN?**

This command initiates and executes a new minimum current measurement, and returns the measured input minimum current of the electronic load.

## Command Syntax

FETCh:CURRent:MIN?

MEASure:CURRent:MIN?

## Parameters

None

## Returned Parameters

<NRf>



## **FETCh:POWer[:DC]?**

This command is used to get the latest measured DC power.

### Command Syntax

FETCh:POWer[:DC]?

### Parameters

None

### Examples

FETC:POW?

### Returned Parameters

<NRf>

## **FETCh:CAPability?**

This command is used to get the latest measured discharging capability.

## **MEASure:CAPability?**

This command initiates and executes a new discharging capability measurement, and returns the measured input discharging capability of the electronic load.

### Command Syntax

FETCh:CAPability?

MEASure:CAPability?

### Parameters

None

### Returned Parameters

<NRf>

## **FETCh:ALLVoltage?**

## **MEASure:ALLVoltage?**

This command is used to read the latest measured voltage of all channels of the instrument.

### Command Syntax

FETCh:ALLVoltage?

MEASure:ALLVoltage?

### Parameters

None

### Examples

FETC:ALLV?

MEAS:ALLV?

### Returned Parameters

<NRf>

## **FETCh:ALLCurrent?**

## **MEASure:ALLCurrent?**

This command is used to read the latest measured current of all channels of the instrument.

### Command Syntax

FETCh:ALLCurrent?

MEASure:ALLCurrent?

### Parameters

None

### Examples

FETC:ALLC?

MEAS:ALLC?

### Returned Parameters

<NRf>

## **MEASure:ALLPower?**

This command is used to read the latest measured power of all channels of the instrument.

## Command Syntax

MEASure:ALLPower?

## Parameters

None

## Examples

MEAS:ALLP?

## Returned Parameters

<NRf>

---

**Note:** Fetch command and measure command both can be used to read all kinds of parameters. The difference is that fetch command is used to read the last measured parameters, but measure command initiates and executes a new measurement, and returns the measured parameter. Fetch command is faster in speed but measure command is more precise.

---

---

## Chapter8 CHANnel Subsystem

---

### CHANnel

These commands select the multiple electronic load channels to which all subsequent channel-specific commands will be directed. If the specified channel number does not exist or is outside the MIN/MAX range, an error code is generated (see appendix C). Refer to the installation section of the User's Guide for more information about channel number assignments.

#### Command Syntax

CHANnel <NR1>

#### Parameters

1-8

#### Examples

CHAN 3

#### Query Syntax

CHANnel?

#### Returned Parameters

<NR1>

### CHANnel:ID?

This command queries the model, serial number and firmware version of the currently selected module, and returns a comma-separated data field.

#### Query Syntax

CHANnel:ID?

#### Parameters

None

#### Returned Parameters

<AARD>	Field	Information
xxxxA		model number
nnnn		serial number or 0

Vxx.xx          firmware revision

### Examples

IT8731, 002001, V1.01

---

## Chapter9 TRACe Subsystem

---

The commands in this subsystem are used to configure and control data storage into the buffer.

### TRACe:CLEAr

This action command is used to clear the buffer of readings. If you do not clear the buffer, a subsequent store will overwrite the old readings. If the subsequent store is aborted before the buffer becomes full, you could end up with some “old” readings still in the buffer.

#### Command Syntax

TRACe:CLEAr

#### Parameters

None

#### Example

STAT:PRES

### TRACe:FREE?

This command is used to read the status of storage memory. After sending this command and addressing the electronic to talk, two values separated by commas are sent to the computer. The first value indicates how many bytes of memory are available, and the second value indicates how many bytes are reserved to store readings.

#### Query Syntax

TRACe:FREE?

#### Returned Parameters

<NR1>, <NR1>

#### Examples

TRAC:FREE?

### TRACe:POINTs

This command is used to specify the size of the buffer.

## Command Syntax

TRACe:POINts <NRf+>

## Parameters

2 to 1000 | MINimum | MAXimum | DEFault

## \*RST Value

1000

## Examples

TRAC:POIN 10

## Query Syntax

TRACe:POINts? [ MINimum | MAXimum | DEFault ]

## Returned Parameters

<NR1>

## Related Commands

TRAC:FEED

# TRACe:FEED

This command is used to select the source of readings to be placed in the buffer. With VOLTage selected, voltage readings are placed in the buffer, TRAC:POIN maximum values is 2000. With CURRent selected, current readings are placed in the buffer, TRAC:POIN maximum values is 2000. With TWO selected, voltage and current are placed in the buffer when storage is performed, TRAC:POIN maximum values is 1000.

## Command Syntax

TRACe:FEED <CRD>

## Parameters

VOLTage | CURRent | TWO

## \*RST Value

TWO

## Examples

TRAC:FEED VOLT

## Query Syntax

TRACe:FEED?

## Returned Parameters

<CRD>

## Related Commands

TRAC:POIN

# TRACe:FEED:CONTRol

This command is used to select the buffer control. With NEVer selected, storage into the buffer is disabled. When NEXT is selected, the storage process starts, fills the buffer and then stops. The buffer size is specified by the :POINTs command.

## Command Syntax

TRACe:FEED:CONTRol <CRD>

## Parameters

NEVer | NEXT

## \*RST Value

NEVer

## Examples

TRAC:FEED:CONT NEXT

## Query Syntax

TRACe:FEED:CONT?

## Returned Parameters

<CRD>

## Related Commands

TRAC:FEED

# TRACe:DATA?

When this command is sent and the electronic load is addressed to talk, all the readings stored in the buffer are sent to the computer.

Before executing this command, you need to send TRIGger[:IMMediate] to



trigger the instrument to enter the data storage state.

### Query Syntax

TRACe:DATA?

### Returned Parameters

{<NR3>}

## TRACe:FILTer

This command is used to select whether the data in buffer is filtered data.

### Command Syntax

TRACe:FILTer[:STATe] <BOOL>

### Parameters

0 | 1 | ON | OFF

### \*RST Value

OFF

### Examples

TRAC:FILT 1

### Query Syntax

TRACe:FILTer[:STATe]?

### Returned Parameters

<NR1>

## TRACe:DELay

This command is used to select the delay time for trigger in buffer.

### Command Syntax

TRACe:DELay <NRf>

### Parameters

0 to 3600s | MINimum | MAXimum | DEFault

### Unit

S (second)

**\*RST Value**

0

**Examples**

TRAC:DEL 1

**Query Syntax**

TRACe:DElay? [MINimum | MAXimum | DEFault]

**Returned Parameters**

&lt;NR3&gt;

## TRACe:TIMer

This command is used to select the interval for timer.

**Command Syntax**

TRACe:TIMer &lt;NRf&gt;

**Parameters**

0.00002 to 3600s | MINimum | MAXimum | DEFault

**Unit**

S (second)

**\*RST Value**

1

**Examples**

TRAC:TIM 0.1

**Query Syntax**

TRACe:TIMer? [MINimum | MAXimum | DEFault]

**Returned Parameters**

&lt;NR3&gt;

---

## Chapter10 SOURce Subsystem

---

These commands control the input of the electronic load. The INPut and OUTPut commands are equivalent. The CURRent, RESistance and VOLTage commands program the actual input current, resistance, and voltage.

### [SOURce:]INPut:ALL

These commands enable or disable all models inputs. The state of a disabled input is a high impedance condition.

#### Command Syntax

```
[SOURce:]INPut:ALL[:STATe] <bool>
```

#### Parameters

```
0 | 1 | OFF | ON
```

#### Examples

```
INP:ALL 1
```

### [SOURce:]INPut

These commands enable or disable the input of the current channel. The state of a disabled input is a high impedance condition.

#### Command Syntax

```
[SOURce:]INPut[:STATe] <bool>
```

#### Parameters

```
0 | 1 | OFF | ON
```

#### \*RST Value

```
OFF
```

#### Examples

```
INP 1
```

#### Query Syntax

```
INPut[:STATe]?
```

## Returned Parameters

0 | 1

## Related Commands

\*RCL \*SAV

# [SOURce:]INPut:CONTRol

This command is used to select the external analog control status of the load.

## Command Syntax

[SOURce:]INPut:CONTRol

## Parameters

INTernal|EXTernal

## \*RST Value

INTernal

## Examples

INP:CONT INT

## Query Syntax

[SOURce:]INPut:CONTRol?

## Returned Parameters

<CRD>

# [SOURce:]INPut:SYNCon

These commands enable or disable to change the electronic load inputs when received INP:ALL command.

## Command Syntax

[SOURce:]INPut:SYNCon[:STATe] <bool>

## Parameters

0 | 1 | OFF | ON

## \*RST Vlaue

ON

### Examples

INP:SYNC 1

### Query Syntax

INPut:SYNCon[:STATe]?

### Returned Parameters

0 | 1

### Related Commands

INP:ALL

## **[SOURce:]INPut:SHORT**

This command programs the specified electronic load module to the maximum current that it can sink in the present operating range.

### Command Syntax

[SOURce:]INPut:SHORT[:STATe] <bool>

### Parameters

0 | 1 | OFF | ON

### Examples

INP:SHOR 1

### Query Syntax

INPut:SHORT:STATe?

### Returned Parameters

0 | 1

### Related Commands

INP

## **[SOURce:]INPut:TIMer**

These commands enable or disable the load on timer.

### Command Syntax

[SOURce:]INPut:TIMer[:STATe] <bool>

**Parameters**

0 | 1 | OFF | ON

**\*RST Value**

OFF

**Examples**

INP:TIM 1

**Query Syntax**

INPut:TIMer[:STATe]?

**Returned Parameters**

0 | 1

**Related Commands**

INP:TIM:DEL

**[SOURce:]INPut:TIMer:DELay**

This command specifies the load on timer.

**Command Syntax**

[SOURce:]INPut:TIMer:DELay &lt;NRf+&gt;

**Parameters**

0.01 to 60000s | MINimum | MAXimum | DEFault

**Unit**

seconds

**\*RST Value**

10

**Examples**

INP:TIM:DEL 5

**Query Syntax**

[SOURce:]INPut:TIMer:DELay?

**Returned Parameters**

&lt;NR3&gt;

## Related Commands

INP:TIM

## [SOURce:]REMOte:SENSe

This command is used to select the remote measure mode for the load.

### Command Syntax

[SOURce:]REMOte:SENSe[:STATe] <BOOL>

### Parameters

0 | 1 | OFF | ON

### \*RST Value

0

### Examples

REM:SENS 0

### Query Syntax

[SOURce:]REMOte:SENSe[:STATe]?

### Returned Parameters

<CRD>

## [SOURce:]FUNction

These equivalent commands select the input regulation mode of the electronic load.

- **CURRent** constant current mode
- **RESistance** constant resistance mode
- **VOLTage** constant voltage mode
- **POWer** constant power mode

### Command Syntax

[SOURce:]FUNction <function>

### Parameters

CURRent | RESistance | VOLTage | POWer

### \*RST Value

CURRent

## Examples

FUNC RES

## Query Syntax

[SOURce:]FUNCTION?

## Returned Parameters

<CRD>

## **[SOURce:]FUNCTION:MODE**

This command determines whether the input regulation mode is controlled by values in a list or by the FUNCTION command setting.

- **FIXed**: The regulation mode is determined by the FUNCTION or MODE command.
- **LIST**: The regulation mode is determined by the active list.

## Command Syntax

[SOURce:]FUNCTION:MODE <mode>

## Parameters

FIXed | LIST

## Returned Parameters

FIXed

## Examples

FUNC:MODE FIX

## Query Syntax

[SOURce:]FUNCTION:MODE?

## Returned Parameters

<CRD>

## Related Commands

FUNC

## **[SOURce:]TRANsient**

This command turns the transient generator on or off.



## Command Syntax

[SOURce:]TRANsient[:STATe] <bool>

## Parameters

0 | 1 | OFF | ON

## \*RST Value

OFF

## Examples

TRAN 1

## Query Syntax

[SOURce:]TRANsient[:STATe]?

## Returned Parameters

0 | 1

## Related Commands

CURR:TRAN:CURR:MODE CURR:TRAN:ALEV

## **[SOURce:]PROTection:CLEar**

This command clears the latch that disables the input when a protection condition such as overvoltage (OV) or overcurrent (OC) is detected. All conditions that generated the fault must be removed before the latch can be cleared. The input is then restored to the state it was in before the fault condition occurred.

## Command Syntax

[SOURce:]PROTection:CLEar

## Parameters

None

## Examples

PROT:CLE

## **[SOURce:]CURREnt**

This command sets the current that the load will regulate when operating in constant current mode.

## Command Syntax

[SOURce:]CURRent[:LEVeI][:IMMediate] <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

A (amperes)

## \*RST Value

MINimum

## Examples

CURR 5    CURR:LEV 0.5

## Query Syntax

[SOURce:]CURRent[:LEVeI][:IMMediate]?

## Returned Parameters

<NR3>

## Related Commands

CURR:RANG

# [SOURce:]CURRent:RANGe

This command sets the current range of the electronic load module. There are two current ranges.

- **High Range**
- **Low Range**

When you program a range value, the load automatically selects the range that corresponds to the value that you program. If the value falls in a region where ranges overlap, the load selects the range with the highest resolution.

---

**NOTE: When this command is executed, the IMMEDIATE, TRANSient, TRIGgered, and SLEW current settings are adjusted as follows:**

**If the existing settings are within the new range: No adjustment is made.**

**If the existing settings are outside the new range: The levels are set to the maximum value of the new range.**

---

## Command Syntax

[SOURce:]CURRent:RANGe <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

A (amperes)

## \*RST Value

MAXimum (high range)

## Examples

SOUR:CURR:RANGE MIN

## Query Syntax

[SOURce:]CURRent:RANGe?

## Returned Parameters

<NR3>

## Related Commands

CURR CURR:SLEW

# **[SOURce:]CURRent:SLEW**

This command sets the slew rate for all programmed changes in the input current level of the electronic load. This command programs both positive and negative going slew rates. MAXimum sets the slew to the fastest possible rate. MINimum sets the slew to the slowest rate.

## Command Syntax

[SOURce:]CURRent:SLEW[:BOTH] <NRf+>

## Parameters

MINimum to MAXimum | MAXimum | MINimum | DEFault

## Unit

A (amps per micro second)

## \*RST Value

MAXimum

## Examples

CURR:SLEW MAX

## Related Commands

CURR:SLEW:NEG

CURR:SLEW:POS

## **[SOURce:]CURRent:SLEW:POSitive**

This command sets the slew rate of the current for positive going transitions. MAXimum sets the slew to the fastest possible rate. MINimum sets the slew to the slowest rate.

## Command Syntax

[SOURce:]CURRent:SLEW:POSitive <NRf+>

## Parameters

MINimum to MAXimum | MAXimum | MINimum | DEFault

## Unit

A (amps per micro second)

## \*RST Value

MAXimum

## Examples

CURR:SLEW:POS MAX

## Query Syntax

[SOURce:]CURRent:SLEW:POSitive?

## Returned Parameters

<NR3>

## Related Commands

CURR:SLEW

## **[SOURce:]CURRent:SLEW:NEGative**

This command sets the slew rate of the current for negative going transitions. MAXimum sets the slew to the fastest possible rate. MINimum sets the slew to the slowest rate.

## Command Syntax

[SOURce:]CURRent:SLEW:NEGative <NRf+>

## Parameters

MINimum to MAXimum | MAXimum | MINimum | DEFault

## Unit

A (amps per micro second)

## \*RST Value

MAXimum

## Examples

CURR:SLEW:NEG MAX

## Query Syntax

[SOURce:]CURRent:SLEW:NEGative?

## Returned Parameters

<NR3>

## Related Commands

CURR:SLEW

## **[SOURce:]CURRent:PROTection[:STATe]**

This command enables or disables the over-current protection feature.

## Command Syntax

[SOURce:]CURRent:PROTection[:STATe] <Bool>

## Parameters

0 | 1 | OFF | ON

## \*RST Value

OFF

## Examples

CURR:PROT 1

## Query Syntax

[SOURce:]CURRent:PROTection[:STATe]?

## Returned Parameters

0 | 1

## Related Commands

CURR:PROT:LEV

# [SOURce:]CURRent:PROTection:LEVel

This command sets the soft current protection level. If the input current exceeds the soft current protection level for the time specified by CURR:PROT:DEL, the input is turned off.

---

NOTE: Use CURR:PROT:DEL to prevent momentary current limit conditions caused by programmed changes from tripping the overcurrent protection.

---

## Command Syntax

[SOURce:]CURRent:PROTection:LEVel <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

A (amperes)

## \*RST Value

MAXimum

## Examples

CURR:PROT:LEV 2

## Query Syntax

[SOURce:]CURRent:PROTection:LEVel?

## Returned Parameters

NR3

## Related Commands

CURR:PROT:DEL

CURR:PROT

## [SOURce:]CURRent:PROTection:DELaY

This command specifies the time that the input current can exceed the protection level before the input is turned off.

### Command Syntax

```
[SOURce:]CURRent:PROTection:DELaY <NR1>
```

### Parameters

0 to 60 seconds. Time parameter needs to be an integer.

### Unit

seconds

### \*RST Value

3

### Examples

```
CURR:PROT:DEL 5
```

### Query Syntax

```
[SOURce:]CURRent:PROTection:DELaY?
```

### Returned Parameters

```
<NR1>
```

### Related Commands

```
CURR:PROT:LEV
```

```
CURR:PROT
```

## [SOURce:]CURRent:TRANsient:MODE

This command selects the operating mode of the transient generator as follows in constant current mode.

- CONTInuous: The transient generator puts out a continuous pulse stream after receipt of a trigger.
- PULSe: The transient generator puts out a single pulse upon receipt of a trigger.
- TOGGle: The transient generator toggles between two levels upon receipt of a trigger.

## Command Syntax

[SOURce:]CURRent:TRANsient:MODE <mode>

## Parameters

CONTInuous | PULSe | TOGGle

## \*RST Value

CONTInuous

## Examples

CURR:TRAN:MODE TOGG

## Query Syntax

[SOURce:]CURRent:TRANsient:MODE?

## Returned Parameters

<CRD>

## Related Commands

CURR:TRAN:ALEV TRAN

# **[SOURce:]CURRent:TRANsient:ALEVel**

# **[SOURce:]CURRent:TRANsient:BLEVel**

This command specifies the transient level of the input current. The transient function switches between the level a and level b.

## Command Syntax

[SOURce:]CURRent:TRANsient:ALEVel <NRf+>

[SOURce:]CURRent:TRANsient:BLEVel <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

A (amperes)

## \*RST Value

ALEVEL MINnum, BLEVel MINnum



## Examples

CURR:TRAN:ALEV 5    CURR:TRAN:BLEV 0.5

## Query Syntax

[SOURce:]CURRent:TRANsient:ALEV?

[SOURce:]CURRent:TRANsient:BLEV?

## Returned Parameters

<NR3>

## Related Commands

CURR:

# **[SOURce:]CURRent:TRANsient:AWIDth**

# **[SOURce:]CURRent:TRANsient:BWIDth**

This command specifies the transient pulse width of the input current.

## Command Syntax

[SOURce:]CURRent:TRANsient:AWIDth <NRf+>

[SOURce:]CURRent:TRANsient:BWIDth <NRf+>

## Parameters

20us-3600s

## Unit

S (second)

## \*RST Value

500uS

## Examples

CURR:TRAN:AWID 0.001    CURR:TRAN:BWID 0.02

## Query Syntax

[SOURce:]CURRent:TRANsient:AWIDth?

[SOURce:]CURRent:TRANsient:BWIDth?

Returned Parameters

<NR3>

## **[SOURce:]CURRent:HIGH**

## **[SOURce:]CURRent:LOW**

This command sets the high and low voltage level when the load is in constant current mode.

Command Syntax

[SOURce:]CURRent:HIGH <NRf+>

[SOURce:]CURRent:LOW <NRf+>

Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

Unit

V (volts)

\*RST Value

MAXimum

MINimum

Examples

CURR:HIGH 5

Query Syntax

[SOURce:]CURRent:HIGH?

[SOURce:]CURRent:LOW?

Returned Parameters

<NR3>

## **[SOURce:]CURRent:TRIGgered**

This command sets the current level that will become active when the next trigger occurs.

Command Syntax

[SOURce:]CURRent[:LEVel]:TRIGgered <NRf+>

**Parameters**

0 to MAXimum | MAXimum | MINimum | DEFault

**Unit**

A (amperes)

**\*RST Value**

MINimum

**Examples**

CURR:TRIG 15

**Related Commands**

CURR

**[SOURce:]VOLTage**

This command sets the voltage that the load will regulate when operating in constant voltage mode.

**Command Syntax**

[SOURce:]VOLTage[:LEVel][:IMMediate] &lt;NRf+&gt;

**Parameters**

MINimum through MAX | MINimum | MAXimum | DEFault

**Unit**

V (volts)

**\*RST Value**

MAXimum

**Examples**

VOLT 5

**Query Syntax**

[SOURce:]VOLTage[:LEVel][:IMMediate]?

**Returned Parameters**

&lt;NR3&gt;

## Related Commands

VOLT:RANG

## [SOURce:]VOLTage:RANGe

This command sets the voltage range of the electronic load module. There are only one voltage ranges.

### Command Syntax

[SOURce:]VOLTage:RANGe <NRf+>

### Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

### Unit

V (volts)

### \*RST Value

MAXimum

### Examples

VOLT:RANG 15

### Query Syntax

[SOURce:]VOLTage:RANGe?

### Returned Parameters

<NR3>

## Related Commands

VOLT

## [SOURce:]VOLTage:RANGe:AUTO[:STATe]

This command sets the voltage auto range state of the electronic load module.

### Command Syntax

[SOURce:]VOLTage:RANGe:AUTO[:STATe] <bool>

### Parameters

0 | 1 | ON | OFF

**\*RST Value**

1

**Examples**

VOLT:RANG:AUTO 1

**Query Syntax**

[SOURce:]VOLTage:RANGe:AUTO[:STATe]?

**Returned Parameters**

&lt;NR1&gt;

**[SOURce:]VOLTage:ON**

This command sets the voltage of sink current on.

**Command Syntax**

[SOURce:]VOLTage[:LEVel]:ON &lt;NRf+&gt;

**Parameters**

0 through MAX | MINimum | MAXimum | DEFault

**Unit**

V (volts)

**\*RST Value**

MINimum

**Examples**

VOLT:ON 5

**Query Syntax**

[SOURce:]VOLTage[:LEVel]:ON?

**Returned Parameters**

&lt;NR3&gt;

**Related Commands**

VOLT:LATCh

## [SOURce:]VOLTage:LATCh

This command sets the action type of Von.

### Command Syntax

```
[SOURce:]VOLTage:LATCh[:STATe] <b>
```

### Parameters

0 | 1 | ON | OFF

### \*RST Value

OFF

### Examples

```
VOLT:LATC 1
```

### Query Syntax

```
[SOURce:]VOLTage:LATCh[:STATe]?
```

### Returned Parameters

0 | 1

### Related Commands

VOLT:ON

## [SOURce:]VOLTage:HIGH

## [SOURce:]VOLTage:LOW

This command set the high and low current limit when the load is in constant voltage mode.

### Command Syntax

```
[SOURce:]VOLTage:HIGH <NRf+>
```

```
[SOURce:]VOLTage:LOW <NRf+>
```

### Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

### Unit

A (amps)

**\*RST Value**

MAXimum

MINimum

**Examples**

VOLT:HIGH 5

**Query Syntax**

[SOURce:]VOLTage:HIGH?

[SOURce:]VOLTage:LOW?

**Returned Parameters**

&lt;NR3&gt;

**[SOURce:]VOLTage:ILIMit <NRf+>**

This command sets limited current value under CV mode.

**Command Syntax**

[SOURce:]VOLTage:ILIMit &lt;NRf+&gt;

**Parameters**

0 through MAX | MINimum | MAXimum

**Examples**

VOLT:ILIM 5

**Query Syntax**

[SOURce:]VOLTage:ILIMit?

**Returned Parameters**

&lt;NR3&gt;

**[SOURce:]VOLTage[:LOOP]:RATE <SLOW|FAST>**

This command sets the CV loop speed.

**Command Syntax**

[SOURce:]VOLTage[:LOOP]:RATE &lt;SLOW|FAST&gt;

**Parameters**

SLOW|FAST

## Examples

VOLT:RATE SLOW

## Query Syntax

[SOURce:]VOLTage[:LOOP]:RATE?

## Returned Parameters

SLOW|FAST

## [SOURce:]VOLTage:TRIGgered

This command sets the voltage level that will become active when the next trigger occurs.

## Command Syntax

[SOURce:]VOLTage[:LEVel]:TRIGgered <NRf+>

## Parameters

MINimum to MAXimum | MAXimum | MINimum | DEFault

## Unit

V (volts)

## \*RST Value

MAXimum

## Examples

VOLT:TRIG 80 VOLT:LEV:TRIG 50

## Related Commands

VOLT

## [SOURce:]RESistance

This command sets the resistance of the load when operating in constant resistance mode.

## Command Syntax

[SOURce:]RESistance[:LEVel][:IMMediate] <NRf+>

## Parameters

MINimum through MAX | MINimum | MAXimum | DEFault



**Unit**

R(ohms)

**\*RST Value**

MAXimum

**Examples**

RES 5 RES:LEV 3.5

**Query Syntax**

[SOURce:]RESistance[:LEVel][:IMMediate]?

**Returned Parameters**

&lt;NR3&gt;

**Related Commands**

RES:RANG

**[SOURce:]RESistance:RANGe**

This command sets the resistance range of the electronic load module.

**Command Syntax**

[SOURce:]RESistance:RANGe &lt;NRf+&gt;

**Parameters**

MINimum through MAX | MINimum | MAXimum | DEFault

**Unit**

R(ohms)

**\*RST Value**

MAXimum (high range)

**Examples**

RES:RANG 15

SOUR:RES:RANGE MIN

**Query Syntax**

[SOURce:]RESistance:RANGe? [ MINimum | MAXimum | DEFault ]

Returned Parameters

<NR3>

## **[SOURce:]RESistance:HIGH**

## **[SOURce:]RESistance:LOW**

This command is used to set the voltage high and low limit determined when the load is in constant resistance.

Command Syntax

[SOURce:]RESistance:HIGH <NRf+>

[SOURce:]RESistance:LOW <NRf+>

Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

Unit

V (volts)

\*RST Value

MAXimum

MINimum

Examples

RES:HIGH 5

Query Syntax

[SOURce:]RESistance:HIGH? [MINimum|MAXimum|DEFault ]

[SOURce:]RESistance:LOW? [MINimum|MAXimum|DEFault ]

Returned Parameters

<NR3>

## **[SOURce:]RESistance:ILIMit <NRf+>**

This command sets limited current value under CR mode.

Command Syntax

[SOURce:]RESistance:ILIMit <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum

## Examples

RES:ILIM 5

## Query Syntax

[SOURce:]RESistance:ILIMit?

## Returned Parameters

<NR3>

## [SOURce:]RESistance:VDRop

This command is used to set the CR mode and the LED Cut-off voltage value when the LED test function is enabled.

## Command Syntax

[SOURce:]RESistance:VDRop <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

V (volts)

## \*RST Value

MINimum

## Examples

RES:VDR 5

## Query Syntax

[SOURce:]RESistance:VDRop? [ MINimum | MAXimum | DEFault ]

## Returned Parameters

<NR3>

## Related Commands

VOLT:VDR

## [SOURce:]RESistance:LED[:STATe]

This command enables or disenables the LED test option in CR mode.

### Command Syntax

[SOURce:]RESistance:LED[:STATe] <b>

### Parameters

0 | 1 | ON | OFF

### \*RST Value

ON

### Examples

RES:LATC 1

### Query Syntax

[SOURce:]RESistance:LED[:STATe]?

### Returned Parameters

0 | 1

### Related Commands

VOLT:ON

## [SOURce:]POWER

This command sets the power of the load when operating in constant power mode.

### Command Syntax

[SOURce:]POWER[:LEVel][:IMMEDIATE] <NRf+>

### Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

### Unit

W (power)

### \*RST Value

MINimum

## Examples

POW 5 POW:LEV 3.5

## Query Syntax

[SOURce:]POWer[:LEVeI][:IMMEDIATE]? [ MINimum | MAXimum | DEFault ]

## Returned Parameters

<NR3>

## Related Commands

POW:RANG

# [SOURce:]POWer:RANGe

This command sets the power range of the electronic load module.

## Command Syntax

[SOURce:]POWer:RANGe <NRf+>

## Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

## Unit

W (power)

## \*RST Value

MAXimum (high range)

## Examples

POW:RANG 15

SOUR:POW:RANGE MIN

## Query Syntax

[SOURce:]POWer:RANGe? [ MINimum | MAXimum | DEFault ]

## Returned Parameters

<NR3>

## [SOURce:]POWer:HIGH

## [SOURce:]POWer:LOW

This command sets the voltage high and low limit determined when in constant power mode.

### Command Syntax

[SOURce:]POWer:HIGH <NRf+>

[SOURce:]POWer:LOW <NRf+>

### Parameters

MINimum through MAX | MINimum | MAXimum | DEFault

### Unit

W (Power)

### \*RST Value

MAXimum

MINimum

### Examples

POW:HIGH 5

### Query Syntax

[SOURce:]POWer:HIGH? [MINimum|MAXimum|DEFault ]

[SOURce:]POWer:LOW? [MINimum|MAXimum|DEFault ]

### Returned Parameters

<NR3>

## [SOURce:]POWer:ILIMit <NRf+>

This command sets limited current value under CW mode.

### Command Syntax

[SOURce:]POWer:ILIMit <NRf+>

### Parameters

0 through MAX | MINimum | MAXimum

## Examples

POW:ILIM 5

## Query Syntax

[SOURce:]POWer:ILIMit?

## Returned Parameters

<NR3>

# [SOURce:]POWer:PROTection[:LEVel]

This command sets the soft power protection level. If the input power exceeds the soft power protection level for the time specified by POW:PROT:DEL, the input is turned off.

---

**NOTE: Use POW:PROT:DEL to prevent momentary power limit conditions caused by programmed changes from tripping the overpower protection.**

---

## Command Syntax

[SOURce:]POWer:PROTection[:LEVel] <NRf+>

## Parameters

0 through MAX | MINimum | MAXimum | DEFault

## Unit

W (power)

## \*RST Value

MAXimum

## Examples

POW:PROT 100

## Query Syntax

[SOURce:]POWer:PROTection[:LEVel]? [ MINimum | MAXimum | DEFault ]

## Returned Parameters

<NR3>

## Related Commands

POW:PROT:DEL

## **[SOURce:]POWer:PROTection:DELaY**

This command specifies the time that the input power can exceed the protection level before the input is turned off.

### Command Syntax

[SOURce:]POWer:PROTection:DELaY <NRf+>

### Parameters

0 to 60 seconds | MINimum | MAXimum | DEFault

### Unit

seconds

### \*RST Value

3

### Examples

POW:PROT:DEL 5

### Query Syntax

[SOURce:]POWer:PROTection:DELaY? [ MINimum | MAXimum | DEFault ]

### Returned Parameters

<NR1>

## Related Commands

POW:PROT

## **[SOURce:]POWer:CONFIg**

This command sets the hard power protection level.

### Command Syntax

[SOURce:]POWer:CONFIg[:LEVe] <NRf+>

### Parameters

0 through MAX | MINimum | MAXimum | DEFault



**Unit**

W (power)

**\*RST Value**

MAXimum

**Examples**

POW:CONFig 100

**Query Syntax**

[SOURce:]POWer:CONFig[:LEVel]?

**Returned Parameters**

&lt;NR3&gt;

**Related Commands**

POW:PROT

**[SOURce:]POWer:TRIGgered**

This command is used to set the on-load power value when the next trigger occurs.

**Command Syntax**

[SOURce:]POWer[:LEVel]:TRIGgered[:AMPLitude] &lt;NRf+&gt;

**Parameters**

MINimum through MAX | MINimum | MAXimum | DEFault

**Unit**

W (power)

**\*RST Value**

MAXimum

**Examples**

POW:TRIG 120 POW:LEV:TRIG 150

**Related Commands**

POW

## List Commands

List commands let you program complex sequences of input changes with rapid, precise timing, and synchronized with trigger signals. Each function for which lists can be generated has a list of values that specify the input at each list step.

## [SOURce:]LIST:RANGe

This command sets the current range for list mode.

### Command Syntax

```
[SOURce:]LIST:RANGe <NRf>
```

### Parameters

MIN through MAX

### Unit

None

### Examples

```
LIST:RANGE 30
```

### Query Syntax

```
[SOURce:]LIST:RANGe?
```

### Returned Parameters

```
<NR3>
```

### Related Commands

```
LIST:LEV
```

## [SOURce:]LIST:COUNT

This command sets the number of times that the list is executed before it is completed. The command accepts parameters in the range 1 through 65535, but 65535 is interpreted as infinity.

### Command Syntax

```
[SOURce:]LIST:COUNT <NRf+>
```

### Parameters

1 to 65535 | MINimum | MAXimum

## Examples

LIST:COUN 3

## Query Syntax

[SOURce:]LIST:COUNT? [ MINimum | MAXimum ]

## Returned Parameters

<NR3>

## Related Commands

LIST:STEP

## **[SOURce:]LIST:STEP**

This command sets the steps of the list.

## Command Syntax

[SOURce:]LIST:STEP <NRf+>

## Parameters

2 to 84 | MINimum | MAXimum

## Examples

LIST:STEP 5

## Query Syntax

[SOURce:]LIST:STEP? [ MINimum | MAXimum ]

## Returned Parameters

<NR3>

## Related Commands

LIST:LEV

## **[SOURce:]LIST:LEVeI**

This command specifies the setting for each list step.

## Command Syntax

[SOURce:]LIST:LEVeI <NR1>, <NRf>

## Parameters

1 to steps, MIN to MAX

## Unit

NONE, NONE

## Examples

LIST:LEV 1, 10LIST:LEV 2, 15.2

## Query Syntax

[SOURce:]LIST:LEV? <NR1>

## Returned Parameters

<NR3>

## Related Commands

LIST:RANG

## **[SOURce:]LIST:SLEW**

This command sets the slew rate for each step. This command programs both positive and negative going slew rates. MAXimum sets the slew to its fastest possible rate. MINimum sets the slew to its slowest rate. LIST:SLEW? returns the number of points programmed.

## Command Syntax

[SOURce:]LIST:SLEW[:BOTH] <NR1> ,<NRf>

## Parameters

1 to steps, MIN to MAX

## Unit

NONE, NONE

## Examples

LIST:SLEW 1, 1.5

LIST:SLEW 2, MAX

## Query Syntax

[SOURce:]LIST:SLEW[:BOTH]? <NR1>

## Returned Parameters

<NR3>

## Related Commands

CURR:SLEW      VOLT:SLEW      RES:SLEW

## [SOURce:]LIST:WIDth

This command sets the sequence of list dwell times. Each value represents the time in seconds that the input will remain at the particular list step point before completing the step. If times exceed 3600s, the input remains at the present level until a trigger sequences the next point in the list. Else At the end of the dwell time, the input automatically changes to the next point in the list.

## Command Syntax

[SOURce:]LIST:WIDth <NR1>, <NRf>

## Parameters

1 to steps, 20uS to max

## Unit

NONE, s (seconds)

## Examples

LIST:WID 1, 0.02      LIST:WID 2, 0.5

## Query Syntax

[SOURce:]LIST:WIDth? <NR1>

## Returned Parameters

<NR3>

## [SOURce:]LIST:SAV

This command stores the present list file of the electronic load to a specified location in memory. Up to 7 files can be stored. file in saved in locations 1-7 are volatile, the data are nonvolatile, the data will be saved when power is removed.

## Command Syntax

[SOURce:]LIST:SAV <NR1>

## Parameters

1 to 7

## Example

LIST:SAV 3

## Related Commands

[SOURce:]LIST:RCL

## **[SOURce:]LIST:RCL**

This command restores a list file that was previously stored in memory with a LIST:SAV command to the specified location.

## Command Syntax

[SOURce:]LIST:RCL <NR1>

## Parameters

1 to 7

## Example

LIST:RCL 3

## Related Commands

[SOURce:]LIST:SAV

## Chapter11 SENSE subsystem

The Sense Subsystem is used to configure and control the measurement functions of the electronic load. A function does not have to be selected before you program its various configurations. A function can be selected any time after it has been programmed. Whenever a programmed function is selected, it assumes the programmed states.

### SENSe:AVERage:COUNT

The command is used to specify the filter count. In general, the filter count is the number of readings that are acquired and stored in the filter buffer for the averaging calculation. The larger the filter count, the more filtering that is performed.

#### Command Syntax

```
SENSe:AVERage:COUNT <NRf+>
```

#### Parameters

2 to 16 | MINimum | MAXimum | DEFault

#### \*RST Value

14

#### Examples

```
SENS:AVER:COUN 10
```

#### Query Syntax

```
SENSe:AVERage:COUNT? [ MINimum | MAXimum | DEFault ]
```

#### Returned Parameters

<NR1>

### SENSe:ACQuire:LINE:FREQ <NR1>

Set the frequency of the PLC.

#### Command Syntax

```
SENSe:ACQuire:LINE:FREQ <NR1>
```

### Parameters

50~60

### Examples

SENS:ACQ:LINE:FREQ 55

### Query Syntax

SENSe:ACQuire:LINE:FREQ?

### Returned Parameters

<NR1>

## **SENSe:ACQuire:NPLCount <NR1>**

Set the number of PLC filter cycles.

### Command Syntax

SENSe:ACQuire:NPLCount <NR1>

### Parameters

1-16

### Examples

SENS:ACQ:NPLC 6

### Query Syntax

SENSe:ACQuire:NPLCount?

### Returned Parameters

<NR1>



---

## Chapter12 OCP Testing Commands

---

The OCP test command is specific to the IT8700P+ modules and is not applicable to other models.

### **[SOURce:]OCP:STATe <bool>**

This command is used to set the OCP test status.

#### Command Syntax

0|1|OFF|ON

#### Parameters

[SOURce:]OCP:STATe <bool>

#### Examples

OCP:STAT ON

#### Query Syntax

OCP:STAT?

#### Returned Parameters

0|1

### **[SOURce:]OCP:VON**

Set the Von voltage value for the OCP test.

#### Command Syntax

[SOURce:]OCP:VON

#### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

#### Examples

OCP:VON 5

#### Query Syntax

[SOURce:]OCP:VON?

Returned Parameters

<NRf>

## **[SOURce:]OCP:DELay**

Set the Von delay value for the OCP test.

Command Syntax

[SOURce:]OCP:DELay

Parameters

0-100s

Examples

OCP:DEL 5

Query Syntax

[SOURce:]OCP:DELay?

Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:RANGe**

Set the current range for the OCP test.

Command Syntax

[SOURce:]OCP[:CURRent]:RANGe

Parameters

MIN~MAX | MINimum | MAXimum | DEFault

Examples

OCP:RANG 5

Query Syntax

[SOURce:]OCP[:CURRent]:RANGe?

Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:START**

This command is used to set the start current of the OCP test.

### Command Syntax

[SOURce:]OCP[:CURRent]:START

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OCP:START 1

### Query Syntax

[SOURce:]OCP[:CURRent]:START?

### Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:END**

This command is used to set the OCP cut-off current.

### Command Syntax

[SOURce:]OCP[:CURRent]:END

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OCP:END 10

### Query Syntax

[SOURce:]OCP[:CURRent]:END?

### Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:INCrement**

Set the step current for the OCP test.

## Command Syntax

[SOURce:]OCP[:CURRent]:INCRe ment

## Parameters

MIN~MAX | MINimum | MAXimum | DEFault

## Examples

OCP:INCRe ment 1

## Query Syntax

[SOURce:]OCP[:CURRent]:INCRe ment?

## Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:WIDTh**

Set the step time for the OCP test.

## Command Syntax

[SOURce:]OCP[:CURRent]:WIDTh

## Parameters

0.00002s - 100s

## Examples

OCP:WIDTh 2

## Query Syntax

[SOURce:]OCP[:CURRent]:WIDTh?

## Returned Parameters

<NRf>

## **[SOURce:]OCP:VOLTage:TRIP**

Set the stop voltage for the OCP test.

## Command Syntax

[SOURce:]OCP:VOLTage:TRIP

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OCP:VOLTage:TRIP 10

### Query Syntax

[SOURce:]OCP:VOLTage:TRIP?

### Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:LIMit[:HIGH]**

Set the current comparison high value for OCP test.

### Command Syntax

[SOURce:]OCP[:CURRent]:LIMit[:HIGH]

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OCP:LIMit 8

### Query Syntax

[SOURce:]OCP[:CURRent]:LIMit[:HIGH]?

### Returned Parameters

<NRf>

## **[SOURce:]OCP[:CURRent]:LIMit:LOW**

Set the current comparison low value for OCP test.

### Command Syntax

[SOURce:]OCP[:CURRent]:LIMit:LOW

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

## Examples

OCP:LIMit:LOW 7.5

## Query Syntax

[SOURce:]OCP[:CURRent]:LIMit:LOW?

## Returned Parameters

<NRf>

## **[SOURce:]OCP:SAVe**

This command is used to save the OCP test file.

## Command Syntax

[SOURce:]OCP:SAVe

## Parameters

1-5

## Examples

OCP:SAVe 1

## Query Syntax

None

## Returned Parameters

None

## **[SOURce:]OCP:RECall**

This command is used to recall the OCP test file.

## Command Syntax

[SOURce:]OCP:RECall

## Parameters

1-5

## Examples

OCP:RECall 1

### Query Syntax

None

### Returned Parameters

None

## **[SOURce:]OCP:RESult?**

This command is used to query the OCP test results.

### Command Syntax

[SOURce:]OCP:RESult?

### Parameters

None

### Examples

OCP:RESult?

### Returned Parameters

PASS|FAULT|STOP|RUNNING

## **[SOURce:]OCP:RESult:CURRent?**

The command queries the current value of the OCP test.

### Command Syntax

[SOURce:]OCP:RESult:CURRent?

### Parameters

None

### Examples

OCP:RESult:CURRent?

### Returned Parameters

<NRf>

## Chapter13 OPP Testing Commands

---

The OPP test command is specific to the IT8700P+ modules and is not applicable to other models.

### **[SOURce:]OPP:STATe <bool>**

This command is used to set the OPP test status.

#### Command Syntax

0|1|OFF|ON

#### Parameters

[SOURce:]OPP:STATe <bool>

#### Examples

OPP:STAT ON

#### Query Syntax

OPP:STAT?

#### Returned Parameters

0|1

### **[SOURce:]OPP:VON**

Set the Von voltage value for the OPP test.

#### Command Syntax

[SOURce:]OPP:VON

#### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

#### Examples

OPP:VON 5

#### Query Syntax

[SOURce:]OPP:VON?



Returned Parameters

<NRf>

## **[SOURce:]OPP:DELay**

Set the Von delay value for the OPP test.

Command Syntax

[SOURce:]OPP:DELay

Parameters

0-100s

Examples

OPP:DEL 5

Query Syntax

[SOURce:]OPP:DELay?

Returned Parameters

<NRf>

## **[SOURce:]OPP[:CURRent]:RANGe**

Set the current range for the OPP test.

Command Syntax

[SOURce:]OPP[:CURRent]:RANGe

Parameters

MIN~MAX | MINimum | MAXimum | DEFault

Examples

OPP:RANG 5

Query Syntax

[SOURce:]OPP[:CURRent]:RANGe?

Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWER]:START**

This command is used to set the start power of the OPP test.

### Command Syntax

[SOURce:]OPP[:POWER]:START

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OPP:START 1

### Query Syntax

[SOURce:]OPP[:POWER]:START?

### Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWER]:END**

This command is used to set the OPP cut-off power.

### Command Syntax

[SOURce:]OPP[:POWER]:END

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OPP:END 10

### Query Syntax

[SOURce:]OPP[:POWER]:END?

### Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWER]:INCRe ment**

Set the step power for the OPP test.

### Command Syntax

[SOURce:]OPP[:POWer]:INCRe ment

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OPP:INCRe ment 1

### Query Syntax

[SOURce:]OPP[:POWer]:INCRe ment?

### Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWer]:WIDTh**

Set the step time for the OPP test.

### Command Syntax

[SOURce:]OPP[:POWer]:WIDTh

### Parameters

0.00002s - 100s

### Examples

OPP:WIDTh 2

### Query Syntax

[SOURce:]OPP[:POWer]:WIDTh?

### Returned Parameters

<NRf>

## **[SOURce:]OPP:VOLTage:TRIP**

Set the stop voltage for the OPP test.

### Command Syntax

[SOURce:]OPP:VOLTage:TRIP

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OPP:VOLTage:TRIP 10

### Query Syntax

[SOURce:]OPP:VOLTage:TRIP?

### Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWer]:LIMit[:HIGH]**

Set the power comparison high value for OPP test.

### Command Syntax

[SOURce:]OPP[:POWer]:LIMit[:HIGH]

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

### Examples

OPP:LIMit 8

### Query Syntax

[SOURce:]OPP[:POWer]:LIMit[:HIGH]?

### Returned Parameters

<NRf>

## **[SOURce:]OPP[:POWer]:LIMit:LOW**

Set the power comparison low value for OPP test.

### Command Syntax

[SOURce:]OPP[:POWer]:LIMit:LOW

### Parameters

MIN~MAX | MINimum | MAXimum | DEFault

## Examples

OPP:LIMit:LOW 7.5

## Query Syntax

[SOURce:]OPP[:POWer]:LIMit:LOW?

## Returned Parameters

<NRf>

## **[SOURce:]OPP:SAVe**

This command is used to save the OPP test file.

## Command Syntax

[SOURce:]OPP:SAVe

## Parameters

1-5

## Examples

OPP:SAVe 1

## Query Syntax

None

## Returned Parameters

None

## **[SOURce:]OPP:RECall**

This command is used to recall the OPP test file.

## Command Syntax

[SOURce:]OPP:RECall

## Parameters

1-5

## Examples

OPP:RECall 1

### Query Syntax

None

### Returned Parameters

None

## **[SOURce:]OPP:RESult?**

This command is used to query the OPP test results.

### Command Syntax

[SOURce:]OPP:RESult?

### Parameters

None

### Examples

OPP:RESult?

### Returned Parameters

PASS|FAULT|STOP|RUNNING

## **[SOURce:]OPP:RESult:POWer?**

The command queries the power value of the OPP test.

### Command Syntax

[SOURce:]OPP:RESult:POWer?

### Parameters

None

### Examples

OPP:RESult:POWer?

### Returned Parameters

<NRf>

## Chapter14 Error Messages

### Error Number List

This appendix gives the error numbers and descriptions that are returned by the electronic load. Error numbers are returned in two ways:

- Error numbers are displayed on the front panel
- Error numbers and messages are read back with the SYSTem:ERRor? query. SYSTem:ERRor? returns the error number into a variable and returns two parameters, an NR1 and a string.

The following table lists the errors that are associated with SCPI syntax errors and interface problems. It also lists the device dependent errors. Information inside the brackets is not part of the standard error message, but is included for clarification. When errors occur, the Standard Event Status register records them in bit 2, 3, 4, or 5:

### Command Errors 100 through 199 (sets Standard Event Status Register bit #5 CME)

Error	Error String
101	device independed error
	Too many numeric suffices in Command Spec
110	No Input Command to parse
114	Numeric suffix is invalid value
116	Invalid value in numeric or channel list, e.g. out of range
117	Invalid number of dimensions in a channel list
120	Parameter of type Numeric Value overflowed its storage
130	Wrong units for parameter
140	Wrong type of parameter(s)
150	Wrong number of parameters
160	Unmatched quotation mark (single/double) in parameters
165	Unmatched bracket
170	Command keywords were not recognized
180	No entry in list to retrieve (number list or channel list)
190	Too many dimensions in entry to be returned in parameters
191	Too many char

Execution Errors –200 through –299 (sets Standard Event Status Register bit #4 EXE)

<b>Error</b>	<b>Error String</b>
–200	Execution error [generic]
–221	Settings conflict [check current device state]
–222	Data out of range [e.g., too large for this device]
–223	Too much data [out of memory; block, string, or expression too long]
–224	Illegal parameter value [device-specific]
–225	Out of memory
–270	Macro error
–272	Macro execution error
–273	Illegal macro label
–276	Macro recursion error
–277	Macro redefinition not allowed

System Errors –300 through –399 (sets Standard Event Status Register bit #3 DDE)

<b>Error</b>	<b>Error String</b>
–310	System error [generic]
–350	Too many errors [errors beyond 9 lost due to queue overflow]

Query Errors "C400 through "C499 (sets Standard Event Status Register bit #2)

<b>Error</b>	<b>Error String</b>
-400	Query error [generic]
-410	Query INTERRUPTED [query followed by DAB or GET before response complete]
-420	Query UNTERMINATED [addressed to talk, incomplete programming message received]
-430	Query DEADLOCKED [too many queries in command string]
-440	Query UNTERMINATED [after indefinite response]

Selftest Errors 0 through 99 (sets Standard Event Status Register bit #3)

<b>Error</b>	<b>Error String</b>
0	No error
1	Module Initialization Lost



2	Mainframe Initialization Lost
3	Module Calibration Lost
4	Non-volatile RAM STATE section checksum failed
5	Non-volatile RAM RST section checksum failed
10	RAM selftest
11	CVDAC selftest 1
12	CVDAC selftest 2
13	CCDAC selftest 1
14	CCDAC selftest 2
15	CRDAC selftest 1
16	CRDAC selftest 2
20	Input Down
40	Flash write failed
41	Flash erase failed
80	Digital I/O selftest error

Device-Dependent Errors 100 through 32767 (sets Standard Event Status Register bit #3)

<b>Error</b>	<b>Error String</b>
213	RS-232 buffer overrun error
216	RS-232 receiver framing error
217	RS-232 receiver parity error
218	RS-232 receiver overrun error
220	Front panel uart overrun
221	Front panel uart framing
222	Front panel uart parity
223	Front panel buffer overrun
224	Front panel timeout
225	Front Crc Check error
226	Front Cmd Error
401	CAL switch prevents calibration
402	CAL password is incorrect
403	CAL not enabled
404	Computed readback cal constants are incorrect

- 405 Computed programming cal constants are incorrect
- 406 Incorrect sequence of calibration commands
- 407 CV or CC status is incorrect for this command
- 603 FETCH of data that was not acquired
- 604 Measurement overrange

## **Contact Us**

Thanks for purchasing ITECH products. In case of any doubts, please contact us as follows:

1. Visit ITECH website: [www.itechate.com](http://www.itechate.com).
2. Select the most convenient contact method for further information.