

Agilent U2000 Series USB Power Sensors

Programming Guide



Agilent Technologies

Notices

© Agilent Technologies, Inc. 2007

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

Manual Part Number

U2000-90411

Edition

First Edition, July 9, 2007

Printed in Malaysia

Agilent Technologies, Inc.
3501 Stevens Creek Blvd.
Santa Clara, CA 95052 USA

Warranty

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

Technology Licenses

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

Restricted Rights Legend

U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like 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** notice until the indicated conditions are fully understood and met.

WARNING

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

General Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other Agilent products. During the warranty period, Agilent will, at its option, either repair or replace products which prove to be defective. The warranty period begins on the date of delivery or on the date of installation if installed by Agilent.

Restricted Rights Legend

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as “commercial computer software” as defined in DFARS 252.227-7013 (Oct 1988), DFARS 252.211-7015 (May 1991), or DFARS 252.227-7014 (Jun 1995), as a “commercial item” as defined in FAR 2.101(a), or as “restricted computer software” as defined in FAR 52.227-19 (Jun 1987) (or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the Agilent standard software agreement for the product involved.

Equipment Operation

Warnings and Cautions

This guide uses warnings and cautions to denote hazards.

WARNING

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

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like 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 notice until the indicated conditions are fully understood and met.

Personal Safety Considerations

This is a Safety Class I product (provided with a protective earthing ground incorporated in the power cord). The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. Any interruption of the protective conductor, inside or outside the instrument, is likely to make the instrument dangerous. Intentional interruption is prohibited. If this instrument is not used as specified, the protection provided by the equipment could be impaired. This instrument must be used in a normal condition (in which all means of protection are intact) only.

No operator serviceable parts inside. Refer servicing to qualified personnel. To prevent electrical shock, do not remove covers. For continued protection against fire hazard, replace the line fuse(s) only with fuses of the same type and rating (for example, normal blow, time delay, etc.). The use of other fuses or material is prohibited.

General Safety Considerations

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Agilent Technologies, Inc. assumes no liability for the customer's failure to comply with these requirements.

WARNING

BEFORE CONNECTING THE POWER SENSOR TO OTHER INSTRUMENTS ensure that all instruments are connected to the protective (earth) ground. Any interruption of the protective earth grounding will cause a potential shock hazard that could result in personal injury.

CAUTION

- Use the device with the cables provided.
 - Repair or service that is not covered in this manual should only be performed by qualified personnels.
-

User Environment

This instrument is designed for indoor use only.

In This Guide ...

- 1 Power Sensor Remote Operation** Chapter 1 describes the parameters that configure the power sensor and helps you determine settings to optimize performance.
- 2 MEASurement Commands** Chapter 2 explains how to use the MEASure group of instructions to acquire data using a set of high level instructions.
- 3 CALCulate Subsystem** Chapter 3 explains how the CALCulate subsystem is used to perform post acquisition data processing.
- 4 CALibration Subsystem** Chapter 4 explains how the CALibration command subsystem is used to zero and calibrate the power sensor.
- 5 FORMat Subsystem** Chapter 5 explains how the FORMat subsystem is used to set a data format for transferring numeric information.
- 6 MEMory Subsystem** Chapter 6 explains how the MEMory command subsystem is used to create, edit and review sensor calibration tables.
- 7 SENSE Subsystem** Chapter 6 explains how the SENSE command subsystem directly affects device specific settings that are used to make measurements.
- 8 SERVICE Subsystem** Chapter 6 explains how the SERVICE command subsystem is used to obtain and set information useful for servicing the power sensor.
- 9 STATus Subsystem** Chapter 7 explains how the STATus command subsystem enables you to examine the status of the power sensor by monitoring the “Device Status Register”, “Operation Status Register” and the “Questionable Status Register”.
- 10 SYSTEM Subsystem** Chapter 8 explains how the SYSTEM command subsystem is used to return error numbers and messages from the power sensor, preset the power sensor, and query the SCPI version.
- 11 TRIGger Subsystem** Chapter 9 explains how the TRIGger command subsystem is used to synchronize device actions with events.

- 12 UNIT Subsystem** Chapter 10 explains how the UNIT command subsystem is used to set the power sensor measurement units to Watts and % (linear), or dBm and dB (logarithmic).
- 13 IEEE 488.2 Command Reference** Chapter 11 explains how the SERVICE command subsystem is used to obtain and set information useful for servicing the power sensor.
- A Calibration Factor Block Layout** Appendix A contains information on the calibration factor block layout for U2000 Series USB power sensors.

List of Figures

- Figure 1-1 Frequency Dependent Offset Tables 18
- Figure 1-2 Typical averaged readings on U2000 Series USB power sensors 25
- Figure 1-3 Averaging Range Hysteresis 25
- Figure 1-4 Limits Checking Results 28
- Figure 1-5 How Measurement are Calculated 35
- Figure 1-6 Generalized Status Register Model 37
- Figure 1-7 Typical Status Register Bit Changes 38
- Figure 1-8 Status System 40
- Figure 1-9 Hierarchical structure of SCPI 52
- Figure 1-10 Format of <character_data> 55
- Figure 1-11 Format of <non-decimal numeric> 57
- Figure 1-12 Format of <NR1> 58
- Figure 1-13 Format of <NR2> 58
- Figure 1-14 Format of <NR3> 59
- Figure 1-15 Format of <string> 60
- Figure 2-1 Measurement Display CALCulate Block Window 67
- Figure 3-1 Measurement Display CALCulate Block Window 84
- Figure 3-2 CALCulate Block 84
- Figure 7-1 Example of Averaged Readings 166
- Figure 10-1 IEEE 488.2 Arbitrary Block Program Data Format 254
- Figure 12-1 Measurement Display UNIT Block Window 290

List of Tables

Table 1-1	MEASure? and CONFigure Preset States	7
Table 1-2	Range of Values for Measurement Limits	29
Table 1-3	Bit Definitions - Status Byte Register	41
Table 1-4	Bit Definitions - Standard Event Register	42
Table 1-5	Bit Definitions - Questionable Status Registers	44
Table 1-6	Bit change conditions for Questionable Status Register	44
Table 1-7	Bit Definitions - Operation Status	45
Table 1-8	Bit change conditions for Operation Status	46
Table 1-9	Bit Definitions - Device Status Register	47
Table 1-10	Bit change conditions for Device Status Register	48
Table 6-1	Frequency and Offset Factor List	148
Table 6-2	Frequency and Offset Factor List	152
Table 9-1	Commands and events affecting Status Register	222
Table 10-1	DEFault: Power Sensor Presets	256
Table 12-1	Measurement Display UNIT Block Window	290
Table 13-1	*ESE Mapping	296
Table 13-2	*ESR? Mapping	298
Table 13-3	*SRE Mapping	305
Table 13-4	*STB? Mapping	307
Table A-19	Calibration Factor Block Layout: U2000 Series USB power sensors	2

Contents

Notices	ii
General Warranty	iii
Restricted Rights Legend	iii
Equipment Operation	iv
General Safety Considerations	v
User Environment	v
In This Guide ...	vi
List of Figures	ix
List of Tables	x

1 Power Sensor Remote Operation

Introduction	3
Configuring the Remote Interface	4
• Interface Selection	4
• USB Configuration	4
Zeroing and Calibrating the U2000 Series USB Power Sensors	5
• Zeroing	5
• Calibration	6
Making Measurement	7
• Using MEASure?	8
• Using the CONFigure Command	10
• Using the Lower Level Commands	15
Using Frequency Dependent Offset Tables	17
• Overview	17
• Selecting a Frequency Dependent Offset Table	22
• Enabling a Frequency Dependent Offset Table	22
• Making the Measurement	22
Setting the Range and Averaging	24
• Averaging	24
• Auto Averaging Mode	24
• Filter Length	26
• Range	26
Setting Offsets	27
• Channel Offsets	27

Contents

Setting Measurement Limits	28
• Setting Limits	28
• Checking for Limit Failures	29
Getting the Best Speed Performance	31
• Measurement Rate	31
• Trigger Mode	31
• Output Format	33
• Units	33
• Command Used	33
• Fast Mode	34
How Measurements are Calculated	35
Status Reporting	36
• The General Status Register Model	36
• The Condition Polling Method	39
• Status Registers	39
• Device Status Register	47
• Using the Operation Complete Commands	49
Saving and Recalling Power Sensor Configurations	50
• How to Save and Recall a Configuration	50
Using Device Clear to Halt Measurements	51
An Introduction to the SCPI Language	52
• Mnemonic Forms	52
• Using a Colon (:)	52
• Using a Semicolon (;)	53
• Using a Comma (,)	53
• Using Whitespace	53
• Using “?” Commands	53
• Using “*” Commands	54
• Syntax Conventions	54
• Syntax Diagram Conventions	54
• SCPI Data Types	55
• Input Message Terminators	61
SCPI Compliance Information	62
Summary of Commands	63

2 MEASurement Commands

MEASurement Commands	66
CONFigure[1]?	69
CONFigure [1] Commands	71
CONFigure[1][:SCALar][:POWer:AC] [<expected_value>[,<resolution>[,<source list>]]]	72
FETCh[1]?	74
FETCh[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	75
READ[1] Commands	77
READ[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	78
MEASure[1] Commands	80
MEASure[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	81

3 CALCulate Subsystem

CALCulate Subsystem	84
CALCulate[1]:FEED[1]<string>	86
CALCulate[1]:LIMit Commands	89
CALCulate[1]:LIMit:CLEar:AUTo <boolean> ONCE	90
CALCulate[1]:LIMit:CLEar[:IMMediate]	92
CALCulate[1]:LIMit:FAIL?	93
CALCulate[1]:LIMit:FCOut?	94
CALCulate[1]:LIMit:LOWer[:DATA] <numeric_value>	96
CALCulate[1]:LIMit:UPPer[:DATA] <numeric_value>	98
CALCulate[1]:LIMit:STATe <boolean>	101
CALCulate[1]:MATH Commands	103
CALCulate[1]:MATH[:EXPRession] <string>	104
CALCulate[1]:MATH[:EXPRession]:CATalog?	106

4 CALibration Subsystem

CALibration Subsystem	108
CALibration[1][:ALL]	109
CALibration[1][:ALL]?	110
CALibration[1]:AUTO [ONCE ON OFF 0 1]	112
CALibration:ZERO:AUTO [ONCE ON OFF 0 1]	114
CALibration[1]:ZERO:TYPE <EXTernal INTernal>	117

5 FORMat Subsystem

- FORMat Subsystem 120
- FORMat[:READings]:BORDER <character_data> 121
- FORMat[:READings][:DATA] <character_data> 123

6 MEMory Subsystem

- MEMory Subsystem 126
- MEMory:CATalog Commands 128
- MEMory:CATalog[:ALL]? 129
- MEMory:CATalog:STATe? 131
- MEMory:CATalog:TABLE? 132
- MEMory:CLEar Commands 134
- MEMory:CLEar[:NAME] <character_data> 135
- MEMory:CLEar:TABLE 137
- MEMory:FREE Commands 138
- MEMory:FREE[:ALL]? 139
- MEMory:FREE:STATe? 140
- MEMory:FREE:TABLE? 141
- MEMory:NStates? 142
- MEMory:STATe Commands 143
- MEMory:STATe:CATalog? 144
- MEMory:STATe:DEFine <character_data>,<numeric_value> 145
- MEMory:TABLE Commands 147
- MEMory:TABLE:FREQuency <numeric_value>{,<numeric_value>} 148
- MEMory:TABLE:FREQuency:POINts? 151
- MEMory:TABLE:GAIN[:MAGNitude] <numeric_value>{,<numeric_value>} 152
- MEMory:TABLE:GAIN[:MAGNitude]:POINts? 155
- MEMory:TABLE:MOVE <character_data>,<character_data> 156
- MEMory:TABLE:SElect <character_data> 158

7 SENSE Subsystem

- [SENSE] Subsystem 160
- SENSE[1]:AVERage Commands 162
- SENSE[1]:AVERage:COUNT <numeric_value> 163
- SENSE[1]:AVERage:COUNT:AUTO <boolean> 166

SENSe[1]:AVERage:SDETECT <boolean>	169
SENSe[1]:AVERage[:STATe] <boolean>	171
SENSe[1]:CORRection:CSET[2]Commands	173
SENSe[1]:CORRection:CSET[2][:SElect] <string>	174
SENSe[1]:CORRection:CSET[2]:STATe <boolean>	176
[SENSe[1]]:CORRection:FDOFFset GAIN4[:INPut][:MAGNitude]? <boolean>	178
[SENSe[1]]:CORRection:GAIN2 Commands	179
[SENSe[1]]:CORRection:GAIN2:STATe <boolean>	180
[SENSe[1]] SENSe2:CORRection:GAIN2[:INPut][:MAGNitude] <numeric_value>	182
SENSe[1]:DETEctor:FUNCTion <character_data>	185
SENSe[1]:FREQuency[:CW :FIXed] <numeric_value>	187
SENSe[1]:MRATE <character_data>	190
SENSe[1]:POWer:AC:RANGe <numeric_value>	193
SENSe[1]:POWer:AC:RANGe:AUTO <boolean>	195
SENSe[1]:TEMPerature? <boolean>	197

8 SERVICE Subsystem

SERVICE Subsystem	200
SERVICE:BIST:TRIGger:LEVel:STATe?	202
SERVICE:OPTion <character_data>	203
SERVICE:SENSor[1]:CDATe <"date">	204
SERVICE:SENSor[1]:CPLace <"place">	205
SERVICE:SENSor[1]:FREQuency:MAXimum?	207
SERVICE:SENSor[1]:FREQuency:MINimum?	208
SERVICE:SENSor[1]:POWer:AVERage:MAXimum?	209
SERVICE:SENSor[1]:POWer:USABLE:MAXimum?	210
SERVICE:SENSor[1]:POWer:USABLE:MINimum?	211
SERVICE:SENSor[1]:RADc?	212
SERVICE:SENSor[1]:SNUMber <"serial_number">	213
SERVICE:SENSor[1]:TNUMber <"tracking_number">	214
SERVICE:SENSor[1]:TYPE?	216
SERVICE:SNUMber <character_data>	217
SERVICE:VERSion:PROcESSor <character_data>	218
SERVICE:VERSion:SYSTem <character_data>	220

9 STATUS Subsystem

- STATUS Subsystem 222
- Status Register Set Commands 224
- Device Status Register Sets 229
- Operation Register Sets 230
- STATUS:OPERation 231
- STATUS:OPERation:CALibrating[:SUMMARY] 232
- STATUS:OPERation:LLFail[:SUMMARY] 233
- STATUS:OPERation:MEASuring[:SUMMARY] 234
- STATUS:OPERation:SENSe[:SUMMARY] 235
- STATUS:OPERation:TRIGger[:SUMMARY] 236
- STATUS:OPERation:ULFail[:SUMMARY] 237
- STATUS:PRESet 238
- Questionable Register Sets 239
- STATUS:QUEStionable 240
- STATUS:QUEStionable:CALibration[:SUMMARY] 241
- STATUS:QUEStionable:POWer[:SUMMARY] 242

10 SYSTEM Subsystem

- SYSTEM Subsystem 246
- SYSTEM:ERRor? 247
- SYSTEM:HELP:HEADers? 254
- SYSTEM:PRESet <character_data> 255
- SYSTEM:VERSion? 258

11 TRIGger Subsystem

- TRIGger Subsystem 260
- ABORt[1] 261
- INITiate Commands 262
- INITiate[1]:CONTInuous <boolean> 263
- INITiate[1][:IMMEDIATE] 265
- INITiate:CONTInuous:ALL <boolean> 266
- INITiate[1]:CONTInuous:SEQuence[1] <boolean> 268
- INITiate[1][:IMMEDIATE]:ALL 270
- INITiate[1][:IMMEDIATE]:SEQuence[1] 271

TRIGger Commands	272
TRIGger[1]:DELay:AUTO <boolean>	273
TRIGger[1][:IMMediate]	275
TRIGger[1]:SOURce BUS EXTernal HOLD IMMediate	276
TRIGger[:SEQuence]:SLOPe <character_data>	279
TRIGger[:SEQuence[1]]:COUNt <numeric_value>	281
TRIGger[:SEQuence[1]]:DELay:AUTO <boolean>	283
TRIGger[:SEQuence[1]]:IMMediate	285
TRIGger[:SEQuence[1]]:SOURce BUS EXTernal HOLD IMMediate	286

12 UNIT Subsystem

UNIT Subsystem	290
UNIT[1]:POWer <amplitude_unit>	291

13 IEEE 488.2 Command Reference

SCPI Compliance Information	294
*CLS	295
*ESE <NRf>	296
*ESR?	298
*IDN?	299
*OPC	300
*OPT?	301
*RCL <NRf>	302
*RST	303
*SAV <NRf>	304
*SRE <NRf>	305
*STB?	307
*TRG	309
*TST?	310
*WAI	311
USBTMC/USB488 Universal Commands	312

Appendix A Calibration Factor Block Layout

Calibration Factor Block Layout	A-2
---------------------------------	-----

Contents



1 Power Sensor Remote Operation

Introduction	3
Configuring the Remote Interface	4
• Interface Selection	4
• USB Configuration	4
Zeroing and Calibrating the U2000 Series USB Power Sensors	5
• Zeroing	5
• Calibration	6
Making Measurement	7
• Using MEASure?	8
• Using the CONFigure Command	10
• Using the Lower Level Commands	15
Using Frequency Dependent Offset Tables	17
• Overview	17
• Editing Frequency Dependent Offset Tables	19
• Selecting a Frequency Dependent Offset Table	22
• Enabling a Frequency Dependent Offset Table	22
• Making the Measurement	22
Setting the Range and Averaging	24
• Averaging	24
• Auto Averaging Mode	24
• Filter Length	26
Range	26



1 Power Sensor Remote Operation

Setting Offsets	27
• Channel Offsets	27
Setting Measurement Limits	28
• Setting Limits	28
• Checking for Limit Failures	29
Getting the Best Speed Performance	31
• Measurement Rate	31
• Trigger Mode	31
• Output Format	33
• Units	33
• Command Used	33
• Fast Mode	34
How Measurements are Calculated	35
Status Reporting	36
• The General Status Register Model	36
• How to Use Register	39
• The Condition Polling Method	39
• Status Registers	39
• Device Status Register	47
• Using the Operation Complete Commands	49
Saving and Recalling Power Sensor Configurations	50
• How to Save and Recall a Configuration	50
Using Device Clear to Halt Measurements	51
An Introduction to the SCPI Language	52
• Mnemonic Forms	52
• Using a Colon (:)	52
• Using a Semicolon (;)	53
• Using a Comma (,)	53
• Using Whitespace	53
• Using "?" Commands	53
• Using "*" Commands	54
• Syntax Conventions	54
• Syntax Diagram Conventions	54
• SCPI Data Types	55
• Input Message Terminators	61
SCPI Compliance Information	62
Summary of Commands	63

This chapter describes the parameters that configure the power sensor and helps you determine settings to optimize performance.

Introduction

This chapter describes the parameters which configure the power sensor and help you determine settings to optimize performance. It contains the following sections:

- [“Configuring the Remote Interface”](#) on page 4.
- [“Zeroing and Calibrating the U2000 Series USB Power Sensors”](#) on page 5.
- [“Making Measurement”](#) on page 7.
- [“Using Frequency Dependent Offset Tables”](#) on page 17.
- [“Setting the Range and Averaging”](#) on page 24.
- [“Setting Offsets”](#) on page 27.
- [“Setting Measurement Limits”](#) on page 28.
- [“Getting the Best Speed Performance”](#) on page 31.
- [“How Measurements are Calculated”](#) on page 35.
- [“Status Reporting”](#) on page 36.
- [“Saving and Recalling Power Sensor Configurations”](#) on page 50.
- [“Using Device Clear to Halt Measurements”](#) on page 51.
- [“An Introduction to the SCPI Language”](#) on page 52.
- [“SCPI Compliance Information”](#) on page 62.
- [“Summary of Commands”](#) on page 63.

Configuring the Remote Interface

This section briefly describes how to configure the USB interface.

NOTE

For more information on configuring the USB remote interface connectivity, refer to the *Agilent Technologies USB/LAN/GPIB Interfaces Connectivity Guide*. If you have installed the *IO Libraries Suite*, you can access the *Connectivity Guide* via the Agilent IO Libraries Control icon. Alternatively, you can access the *Connectivity Guide* via the Web at www.agilent.com/find/connectivity.

Interface Selection

You can choose to control the power sensor using the USB interface.

USB Configuration

The USB interface requires no front panel or remote configuration.

The USB address cannot be changed - it is set at the factory and is unique for each power sensor.

NOTE

Before connecting the USB cable, make sure that I/O software is installed on your computer.

NOTE

For more information about *Agilent IO Libraries* software refer to the *Connectivity Guide*. If you have installed other I/O Software, refer to documentation that accompanies the software.

Zeroing and Calibrating the U2000 Series USB Power Sensors

The U2000 Series USB power sensors do not need manual calibration and zero routines performed. These are performed without removing the power sensor from the source, which is known as internal zeroing. With internal zeroing of U2000 Series USB power sensors, there is no need to disconnect the sensor or power-off the device-under-test (DUT). The U2000 Series does not require 50 MHz reference signal calibration, thus allowing factory calibration for ensuring measurement accuracy. However, users are recommended to perform external zeroing for input signals below -30 dBm for best accuracy.

Zeroing

Zeroing adjusts the power sensor for a zero power reading.

The command `CALibration[1]:ZERO:AUTO [ONCE|ON|OFF|0|1]` causes the power sensor to perform its zeroing routine when enabled. This adjusts the power sensor for a zero power reading.

`1|ON` can only be used for internal zeroing. When `1|ON` is enabled the zero is maintained by a combination of zero *on-the-fly* for measurements and temperature compensation.

Zeroing of the power sensor happens automatically:

- When a 5 °C change in temperature occurs
- When you change the power sensor
- Every 24 hours

To perform external zeroing, the command `CALibration[1]:ZERO:TYPE EXTERNAL` is used, following by `CALibration[1]:ZERO:AUTO ONCE` to start the zeroing process.

Calibration

There is no calibration needed in U2000 Series USB power sensors. The command `CALibration[1]:AUTO ONCE` shows the code compatibility with `CALibration[1]:ZERO:AUTO ONCE`. Users are not required to modify their source code which is used in other power meters or sensors as `CALibration[1]:AUTO ONCE` supports backward compatibility.

Calibration Sequence

This feature allows you to perform a complete calibration sequence with a single query. The query is:

```
CALibration[1][:ALL]?
```

The complete calibration sequence consists of:

- Zeroing and calibrating the power sensor (`CALibration[1]:ZERO:AUTO ONCE` or `CALibration[1]:AUTO ONCE`)

The query enters a number into the output buffer when the sequence is complete. If the result is 0 the sequence was successful. If the result is 1 the sequence failed. Refer to “[CALibration\[1\]\[:ALL\]?](#)” on page 110 for further information.

NOTE

The `CALibration[1][:ALL]` command is identical to the `CALibration[1][:ALL]?` query except that no number is returned to indicate the outcome of the sequence. You can examine the **Questionable Status Register** or the error queue to discover if the sequence has passed or failed. Refer to “[Status Reporting](#)” on page 36 for further information.

Making Measurement

The MEASure? and CONFigure commands provide a straight-forward method to program the power sensor for measurements. You can select the measurement's expected power level and resolution all in one command. The power sensor automatically presets other measurement parameters to default values as shown in [Table 1-1](#) below.

Table 1-1 MEASure? and CONFigure Preset States

Command	MEASure? and CONFigure Setting
Trigger source (TRIGger:SOURCE)	Immediate
Filter ([SENSe[1]]:AVERage:COUNT:AUTO)	On
Filter state([SENSe[1]]:AVERage:STATE)	On
Trigger cycle (INITiate:CONTinuous)	Off
TriggerDelay (TRIGger:DELay:AUTO)	On

An alternative method to program the power sensor is to use the lower level commands. The advantage of using the lower level commands over the CONFigure command is that they give you more precise control of the power sensor. As shown in [Table 1-1](#), the CONFigure command presets various states in the power sensor. It may be likely that you do not want to preset these states. Refer to [“Using the Lower Level Commands”](#) on page 15 for further information.

Using MEASure?

The simplest way to program the power sensor for measurements is by using the MEASure? query. However, this command does not offer much flexibility. When you execute the command, the power sensor selects the best settings for the requested configuration and immediately performs the measurement. You cannot change any settings (other than the expected power value and resolution) before the measurement is taken. This means you cannot fine tune the measurement, for example, you cannot change the filter length. To make more flexible and accurate measurements use the CONFIGure command. The measurement results are sent to the output buffer. MEASure? is a compound command which is equivalent to an ABORT, followed by a CONFIGure, followed by a READ?.

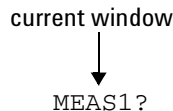
MEASure? Examples

The following commands show a few examples of how to use the MEASure? query to make a measurement. It is advisable to read through these examples in order as they become increasingly more detailed. These examples configure the power sensor for a measurement (as described in each individual example), automatically place the power sensor in the “wait-for-trigger” state, internally trigger the power sensor to take one reading, and then sends the reading to the output buffer.

These examples give an overview of the MEASure? query. For further information on the MEASure? commands refer to the section “MEASurement Commands” on page 66.

Example 1 - The Simplest Method

The following commands show the simplest method of making measurements. Using MEAS1? results in the current window measurement.




Example 2 - Specifying the Source List Parameter

The MEASure command has three optional parameters, an expected power

value, a resolution and a source list. These parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder.

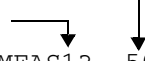
The expected power and resolution parameters are defaulted, leaving them at their current settings. The measurement is carried out on the current window.

current window

 MEAS1? DEF, DEF, (@1)

Example 3 - Specifying the Expected Power Parameter

The previous example details the three optional parameters which can be used with the MEASure? command. The first optional parameter is used to enter an expected power value. The value entered determines which of the power sensor's two ranges is used for the measurement. If the current setting of the power sensor's range is no longer valid for the new measurement, specifying the expected power value decreases the time taken to obtain a result.

The following example uses the expected value parameter to specify a value of -50 dBm. This selects the power sensor's lower range (refer to "Range" on page 26 for details of the range breaks). The resolution parameter is defaulted, leaving it at its current setting.

specifies expected power value
 current window 
 MEAS1? -50, DEF, (@1)

Example 4 - Specifying the Resolution Parameter

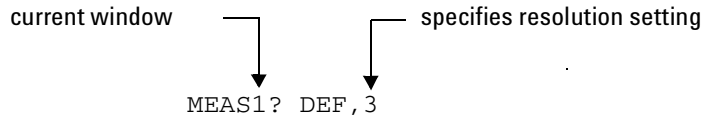
The previous examples detailed the use of the expected value and source list parameters. The resolution parameter is used to set the resolution of the current window. This parameter does not affect the resolution of the data, however it does affect the auto averaging setting (refer to Figure 1-3).

The following example uses the resolution parameter to specify a resolution setting of 3. This setting represents 3 significant digits if the measurement suffix is W or %, and 0.01 dB if the suffix is dB or dBm.

Refer to [Chapter 2, “MEASurement Commands”](#) on page 65, for further details on the resolution parameter. The expected power and source list parameters are defaulted in the example. The expected power value remains unchanged at its current setting. Note that as the source list parameter is the last specified parameter you do not have to specify DEF. The measurement is carried out on the current window.

current window specifies resolution setting

MEAS1? DEF, 3



Using the CONFigure Command

When you execute this command, the power sensor presets the optimum settings for the requested configuration (like the MEASure? query). However, the measurement is not automatically started and you can change measurement parameters before making measurements. This allows you to change the power sensor’s configuration from the preset conditions. The power sensor offers a variety of low-level commands in the SENSE, CALCulate, and TRIGger subsystems. For example, if you want to change the averaging use the [SENSE[1]]:AVERage:COUNT command.

Use the INITiate or READ? query to initiate the measurement.

Using READ?

CONFigure does not take the measurement. One method of obtaining a result is to use the READ? query. The READ? query takes the measurement using the parameters set by the CONFigure command then sends the reading to the output buffer. Using the READ? query obtains new data.

Using INITiate and FETCh?

CONFigure does not take the measurement. One method of obtaining the result is to use the INITiate and FETCh? commands. The INITiate command causes the measurement to be taken. The FETCh? query retrieves a reading when the measurement is complete, and sends the reading to the output buffer. FETCh? can be used to display the measurement results in a number of different formats without taking fresh data for each measurement.

CONFigure Examples

The following program segments show how to use the commands READ?, INITiate and FETCh? and CONFigure to make measurements.

It is advisable to read through these examples in order as they become increasingly more detailed.

These examples give an overview of the CONFigure command. For further information on the CONFigure commands refer to [Chapter 2](#), “MEASurement Commands”.

Example 1 - The Simplest Method

The following program segments show the simplest method of querying the current window’s measurement results respectively.

Using READ?

```
*RST          Reset instrument
CONF1        Configure current window - defaults to a measurement
READ1?      Take current window measurement
```

Using INITiate and FETCh?

```
*RST          Reset instrument
CONF1        Configure current window - defaults to a measurement
INIT1        Causes a measurement
FETC1?       Retrieves the current window’s measurement
```

Example 2 - Specifying the Source List Parameter

The CONFigure and READ? commands have three optional parameters, an expected power value, a resolution and a source list. These parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder.

The following examples use the source list parameter to specify the measurement. The expected power and resolution parameters are defaulted, leaving them at their current settings. The measurement is carried out on the current window.

Although the READ? and FETCh? queries have three optional parameters it

1 Power Sensor Remote Operation

is not necessary to define them as shown in these examples. If they are defined they must be identical to those defined in the `CONFigure` command otherwise an error occurs.

Using `READ?`

<code>ABOR1</code>	<i>Aborts measurement</i>
<code>CONF1 DEF,DEF, (@1)</code>	<i>Configures the current window to make a measurement using the current expected power and resolution settings</i>
<code>READ1?</code>	<i>Takes the current window's measurement</i>

Using `INITiate` and `FETCh?`

<code>ABOR1</code>	<i>Aborts measurement</i>
<code>CONF1 DEF,DEF, (@1)</code>	<i>Configures the current window to make a measurement using the current expected power and resolution settings</i>
<code>INIT1</code>	<i>Causes a measurement</i>
<code>FETC1? DEF,DEF, (@1)</code>	<i>Retrieves the current window's measurement</i>

Example 3 - Specifying the Expected Power Parameter

The previous example details the three optional parameters which can be used with the `CONFigure` and `READ?` commands. The first optional parameter is used to enter an expected power value. The value entered determines which of the power sensor's two ranges is used for the measurement. If the current setting of the power sensor's range is no longer valid for the new measurement, specifying the expected power value decreases the time taken to obtain a result.

The following example uses the expected value parameter to specify a value of -50 dBm. This selects the power sensor's lower range (refer to "[Range](#)" on page 26 for details of the range breaks). The resolution parameter is defaulted, leaving it at its current setting.

Using READ?

ABOR1	<i>Aborts current measurement</i>
CONF1 -50,DEF, (@2)	<i>Configures the current measurement to use an expected power of -50 dBm and the current resolution setting</i>
READ1?	<i>Takes the current measurement</i>

Some fine tuning of measurements can be performed using the CONFigure and READ? commands. For example, in the above program segment some fine tuning can be performed by setting the filter length to 1024 and the trigger delay off.

```

1 ABOR1
2 CONF1 -50,DEF, (@1)
3 SENS1:AVER:COUN 1024
4 TRIG1:DEL:AUTO OFF
5 READ1?

```

Using INITiate and FETCh?

ABOR1	<i>Aborts current measurement</i>
CONF1 -50,DEF, (@1)	<i>Configures the current measurement to use an expected power of -50 dBm and the current resolution setting</i>
INIT1	<i>Causes a measurement</i>
FETC1? -50,DEF, (@1)	<i>Retrieves the current measurement</i>

Some fine tuning of measurements can be carried out using the CONFigure command and INITiate and FETCh? commands. For example, in the above program segment some fine tuning can be carried out by setting the filter length to 1024 and the trigger delay off.

```

1 ABOR1
2 CONF1 -50,DEF, (@1)
3 SENS1:AVER:COUN 1024

```

1 Power Sensor Remote Operation

```
4 TRIG1:DEL:AUTO OFF
5 INIT1
6 FETC1? -50,DEF,(@1)
```

Example 4 - Specifying the Resolution Parameter

The previous examples detailed the use of the expected value and source list parameters. The resolution parameter is used to set the resolution of the measurement. This parameter does not affect the resolution of the data, however it does affect the auto averaging setting (refer to [Figure 1-3](#) on page 25).

The following example uses the resolution parameter to specify a resolution setting of 3. This setting represents 3 significant digits if the measurement suffix is W or %, and 0.01 dB if the suffix is dB or dBm (for further details on the resolution parameter refer to the commands in [Chapter 2](#), “MEASUREMENT Commands”). Also, in this example the expected power and source list parameters are defaulted. The expected power value is left unchanged at its current setting. Note that as the source list parameter is the last specified parameter you do not have to specify DEF.

Using READ?

ABOR1	<i>Aborts current measurement</i>
CONF1 DEF,3	<i>Configures the current measurement to use the current setting of the expected power and source list and a resolution setting of 3</i>
READ1?	<i>Takes the current measurement. This is a measurement depending on current window setup.</i>

Some fine tuning of the above program segment can be carried out for example, by setting the trigger delay off. The following program segment assumes that it is being measured in the measurement.

```
1 ABOR1
2 CONF1 DEF,3
3 TRIG1:DEL:AUTO OFF
4 READ1?
```


Using INITiate and FETCh?

The following program segment assumes that it is being measured on the current window.

ABOR1	<i>Aborts current measurement</i>
CONF1 DEF,3	<i>Configures the current measurement to use the current setting of the expected power and source list and a resolution setting of 3</i>
INIT1	<i>Causes a measurement</i>
FETC1? DEF,3	<i>Retrieves the current measurement</i>

Some fine tuning of the above program segment can be carried out for example, by setting the trigger delay off.

```

1 ABOR1
2 CONF1 DEF,3
3 TRIG1:DEL:AUTO OFF
4 INIT1:IMM
5 FETC1? DEF,3

```

Using the Lower Level Commands

An alternative method of making measurements is to use the lower level commands to set up the expected range and resolution. This can be done using the following commands:

```
[SENSe[1]]:POWer:AC:RANGe
```

The measurement type can be set using the following commands in the CALCulate subsystem:

```
CALCulate[1]:MATH[:EXPRession]
```

The advantage of using the lower level commands over the CONFIGure command is that they give you more precise control of the power sensor. As shown in [Table 1-1](#) the CONFIGure command presets various states in

1 Power Sensor Remote Operation

the power sensor. It may be likely that you do not want to preset these states.

Example

The following example sets the expected power value to -50 dBm and the resolution setting to 3 using the lower level commands.

ABOR1	<i>Aborts current measurement</i>
CALC1:MATH:EXPR " (SENS1) "	<i>Displays current measurement</i>
SENS1:POW:AC:RANG -50	<i>Sets lower range</i>
INIT1	<i>Causes a measurement</i>
FETC1?	<i>Retrieves the current measurement</i>

Using Frequency Dependent Offset Tables

This section describes how to use frequency dependent offset tables. These tables give you the ability to compensate for frequency effects in your test setup.

Overview

If the `[SENSe[1]]:CORRection:CSET2:STATE` command is OFF, the frequency dependent offset tables are not used. When `[SENSe[1]]:CORRection:CSET2:STATE` is ON, the frequency dependent offset tables are used, providing you with a quick and convenient method of compensating for your external test setup over a range of frequencies. Note that when selected, frequency dependent offset correction is IN ADDITION to any correction applied for sensor frequency response. The power sensor is capable of storing 10 frequency dependent offset tables of 80 frequency points each.

To use frequency dependent offset tables you:

- 1 Edit a frequency dependent offset table if necessary.
- 2 Select the frequency dependent offset table.
- 3 Enable the frequency dependent offset table.
- 4 Zero and calibrate the power sensor.
- 5 Specify the frequency of the signal you want to measure. The required offset is automatically set by the power sensor from the frequency dependent offset table.
- 6 Make the measurement.

Figure 1-1 illustrates how frequency dependent offset tables operate.

1 Power Sensor Remote Operation

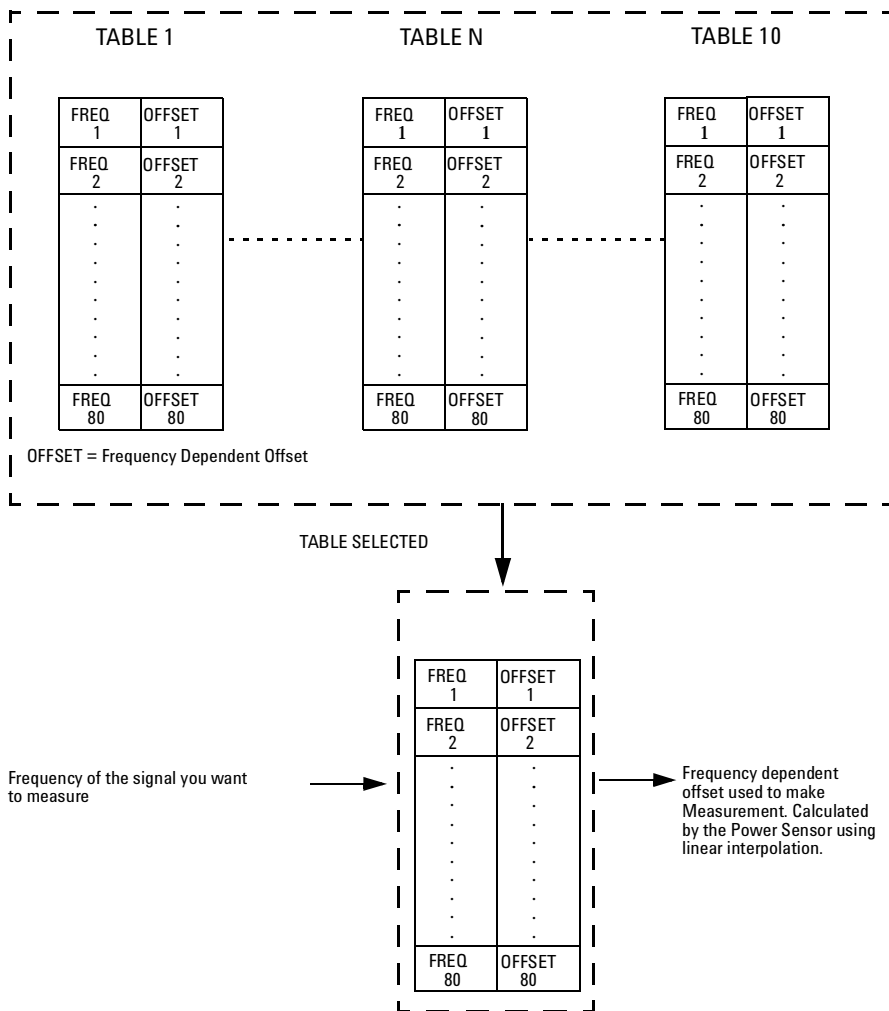


Figure 1-1 Frequency Dependent Offset Tables

Editing Frequency Dependent Offset Tables

It is not possible to create any additional frequency dependent offset tables. However, the 10 existing tables can be edited using the MEMory subsystem. To do this:

- 1 Select one of the existing tables using:

```
MEMory:TABLE:SElect <string>
```

For information on naming frequency dependent offset tables see “[Naming Frequency Dependent Offset Tables](#)” on page 21. For information on the current names which you can select refer to “[Listing the Frequency Dependent Offset Table Names](#)” on page 20.

- 2 Enter the frequency data using:

```
MEMory:TABLE:FREquency <numeric_value> {,<numeric_value>}
```

- 3 Enter the offset factors as shown in the table below using:

```
MEMory:TABLE:GAIN <numeric_value> {,<numeric_value>}
```

Frequency	Offset
Frequency 1	Offset 1
Frequency 2	Offset 2
"	"
Frequency n	Offset n

- 4 If required, rename the frequency dependent offset table using:
MEMory:TABLE:MOVE <string>,<string>. The first <string> parameter identifies the existing table name, and the second identifies the new table name.

NOTE

The legal frequency suffix multipliers are any of the IEEE suffix multipliers, for example, KHZ, MHZ, and GHZ. If no units are specified the power sensor assumes the data is Hz.

PCT is the only legal unit for offset factors and can be omitted.

The frequency and offset data must be within range. Refer to the individual commands in [Chapter 4](#) for their specified ranges.

Any offset values entered into the table should exclude the effect of the sensor. Characterization of the test setup independently of the sensor allows the same table to be used with any sensor.

Ensure that the frequency points you use cover the frequency range of the signals you want to measure. If you measure a signal with a frequency outside the frequency range defined in the frequency dependent offset table, then the power sensor uses the highest or lowest frequency point in the table to calculate the offset.

To make subsequent editing of a frequency dependent offset table simpler, it is recommended that you retain a copy of your data in a program.

Listing the Frequency Dependent Offset Table Names

To list the frequency dependent offset tables currently stored in the power sensor, use the following command:

```
MEMory:CATalog:TABLE?
```

The power sensor returns the data in the form of two numeric parameters and a string list representing all stored tables.

- `<numeric_value>,<numeric_value>{,<string>}`
The first numeric parameter indicates the amount of memory, in bytes, used for storage of tables. The second parameter indicates the memory, in bytes, available for tables.

Each string parameter returned indicates the name, type and size of a stored frequency dependent offset table:

- `<string>,<type>,<size>`
The `<string>`, `<type>` and `<size>` are all character data. The `<type>` is always `TABL`. The `<size>` is displayed in bytes.

For example, a sample of the response may look like:

```
560,8020,"Offset_1,TABL,220","Offset_2,TABL,340" . . . .
```

Naming Frequency Dependent Offset Tables

To rename a frequency dependent offset table use:

```
MEMory:TABLE:MOVE <string>, <string>
```

The first <string> parameter identifies the existing table name, and the second identifies the new table name.

The following rules apply to frequency dependent offset table names:

- 1 Table names use a maximum of 12 characters.
- 2 All characters must be upper or lower case alphabetic characters, or numeric (0-9), or an underscore (_).

No spaces are allowed in the name.

Reviewing Table Data

To review the data stored in a frequency dependent offset table, use the following commands:

```
MEMory:TABLE:SElect "Offset1"
```

Select the frequency dependent offset table named "Offset1".

```
MEMory:TABLE:SElect?
```

Query command which returns the name of the currently selected table.

```
MEMory:TABLE:FREQuency:POINTs?
```

Query command which returns the number of stored frequency points.

```
MEMory:TABLE:FREQuency?
```

Query command which returns the frequencies stored in the frequency dependent offset table (in Hz).

```
MEMory:TABLE:GAIN[:MAGNitude]:POINTs?
```

Query command which returns the number of offset factor points stored in the frequency dependent offset table.

```
MEMory:TABLE:GAIN[:MAGNitude]?
```

Query command which returns the offset factors stored in the frequency dependent offset table.

Modifying Data

If you need to modify the frequency and offset factor data stored in a frequency dependent offset table you need to resend the complete data lists.

If you have retained the original data in a program, edit the program and resend the data.

Selecting a Frequency Dependent Offset Table

After you have created the frequency dependent offset table, you can select it using the following command:

```
[SENSe[1]]:CORRection:CSET2[:SELEct] <string>
```

To find out which frequency dependent offset table is currently selected, use the query:

```
[SENSe[1]]:CORRection:CSET2[:SELEct]?
```

Enabling a Frequency Dependent Offset Table

To enable the frequency dependent offset table, use the following command:

```
[SENSe[1]]:CORRection:CSET2:STATe ON
```

If you set [SENSe[1]]:CORRection:CSET2:STATe to ON and no frequency dependent offset table is selected error -221, “Settings conflict” occurs.

Making the Measurement

To make the power measurement, set the power sensor for the frequency of the signal you want to measure. The power sensor automatically sets the offset factor. Use either the INITiate, FETCh? or the READ? query to initiate the measurement as shown in the following program segments:

INITiate Example

```
ABORt1  
CONFIgure1:POWer:AC DEF,1,(@1)  
SENS1:CORR:CSET2:SEL "Offset1"  
SENS1:CORR:CSET2:STAT ON
```



```

SENS1:FREQuency 500KHZ
INITiate1:IMMediate
FETCh1?

```

READ? Example

```

ABORT1
CONFigure1:POWer:AC DEF,2,(@1)
SENS1:CORR:CSET2:SEL "Offset1"
SENS1:CORR:CSET2:STAT ON
SENS1:FREQuency 500KHZ
READ1?

```

NOTE

If the measurement frequency does not correspond directly to a frequency in the frequency dependent offset table, the power sensor calculates the offset using linear interpolation.

If you enter a frequency outside the frequency range defined in the frequency dependent offset table, then the power sensor uses the highest or lowest frequency point in the table to set the offset.

To find out the value of the offset being used by the power sensor to make a measurement, use the query command:

```
[SENSe[1]]:CORRection:GAIN4|FDOFfset[:INPut][MAGNITUDE]?
```

The response may be an interpolated value.

Setting the Range and Averaging

This section provides an overview of setting the range, resolution and averaging. For more detailed information about these features refer to the individual commands in [Chapter 7](#), “SENSe Subsystem”.

Averaging

The power sensor has a digital filter to average power readings. The number of readings averaged can range from 1 to 2048. This filter is used to reduce noise, obtain the desired resolution and to reduce the jitter in the measurement results. However, the time to take the measurement is increased. You can select the filter length or you can set the power sensor to auto filter mode. To enable and disable averaging use the following command:

```
[SENSe[1]]:AVERage[:STATe] <boolean>
```

Auto Averaging Mode

To enable and disable auto filter mode, use the following command:

```
[SENSe[1]]:AVERage:COUNT:AUTO <boolean>
```

When the auto filter mode is enabled, the power sensor automatically sets the number of readings averaged together to satisfy the filtering requirements for most power measurements. The number of readings averaged together depends on the resolution and the power level currently being measured. [Figure 1-3](#) illustrates part of the power sensor dynamic range hysteresis.

U2000/1/2/4A	Maximum Sensor Power	Resolution Setting			
		1	2	3	4
-35 dBm	↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓	1	1	16	1024
-38 dBm		1	1	1024	1024
-45 dBm		1	1024	1024	1024
-55 dBm		128	1024	1024	1024
-60 dBm		512	1024	1024	1024
-15 dBm		1	1	1	1
-23 dBm		1	1	1	1024
-33 dBm		1	1	256	1024
-38 dBm		1	1	512	1024
Minimum Sensor Power		1	1	1	1

Sensor Dynamic Range: ↑ Range 1 ↓ Range 2 ↓
 Number of Averages: ↓

Figure 1-2 Typical averaged readings on U2000 Series USB power sensors

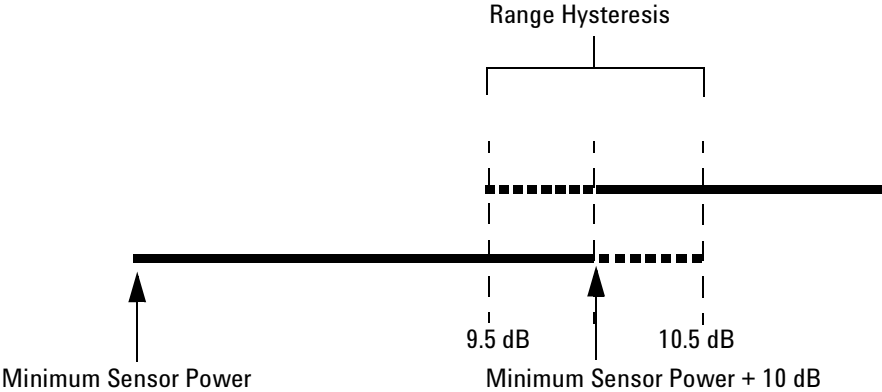


Figure 1-3 Averaging Range Hysteresis

Filter Length

You specify the filter length using the following command:

```
[SENSE[1]]:AVERAge:COUNT <numeric_value>
```

The range of values for the filter length is 1 to 2048. Specifying this command disables automatic filter length selection. Increasing the value of the filter length reduces measurement noise. However, the time to take the measurement is increased.

Range

With an U2000 Series USB Power Sensor the range can be set either automatically or manually. Use autoranging when you are not sure of the power level you will be measuring.

Setting the Range

To set the range manually use the following command:

```
[SENSE[1]]:POWer:AC:RANGe <numeric_value>
```

If the <numeric_value> is set to:

- 0, the sensor's lower range is selected.
- 1, the sensor's upper range is selected.

For detailed on the range limits of U2000 Series USB power sensors, refer to the *U2000 Series USB Power Sensors Operating and Service Guide*.

For further information on this command refer to [page 193](#).

To enable autoranging use the following command:

```
[SENSE[1]]:POWer:AC:RANGe:AUTO ON
```

Use autoranging when you are not sure of the power level you will be measuring.

Setting Offsets

Channel Offsets

The power sensor can be configured to compensate for signal loss or gain in your test setup (for example, to compensate for the loss of a 10 dB attenuator). You use the `SENSE` command subsystem to configure the power sensor. Gain and loss correction are a coupled system. If you enter an offset value the state is automatically enabled. However it can be enabled and disabled using either the

```
[SENSE[1]]:CORREction:GAIN2:STATE.
```

NOTE

To enter a LOSS value, you can enter a negative value in the command

```
[SENSE[1]]:CORREction:GAIN2[:INPut][:MAGNitude]<numeric_value>.
```

Setting Measurement Limits

You can configure the power sensor to detect when a measurement is outside of a predefined upper and/or lower limit value.

Limits are measurement display line based and can be applied to power measurement.

Setting Limits

The power sensor can be configured to verify the power being measured against an upper and/or lower limit value. The range of values that can be set for lower and upper limits is -150.00 dBm to $+230.00$ dBm. The default upper limit is $+90.00$ dBm and the default lower limit is -90.00 dBm.

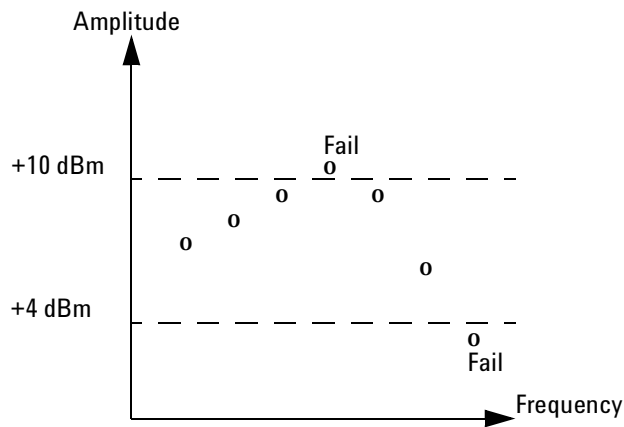


Figure 1-4 Limits Checking Results

Setting Limits (Check Header - same as previous)

The power sensor can be configured to verify the current measurement in any measurement line against predefined upper and/or lower limit values. The range of values that can be set for the upper and lower limits and the default values depends on the measurement units in the currently measurement line - see [Table 1-2](#).

Table 1-2 Range of Values for Measurement Limits

Window Units			Default	
	Maximum	Minimum	Maximum	Minimum
dB	+200 dB	-180 dB	60 dB	-120 dB
dBm	+230 dBm	-150 dBm	90 dBm	-90 dBm
%	999.9 X%	100.0 a%	100.0 M%	100.0 p%
W	100.000 XW	1.000 aW	1.000 MW	1.000 pW

Checking for Limit Failures

There is a way to check for limit failures:

- Use the `CALCulate[1]:LIMit:FAIL?` and `CALCulate[1]:LIMit:FCOut?` commands for measurement limits

Using CALCulate

Using `CALCulate` to check the measurement limit failures in [Figure 1-4](#) would return the following results:

`CALCulate[1]:LIMit:FAIL?`

Returns 1 if there has been 1 or more limit failures or 0 if there have been no limit failures. In this case 1 is returned.

`CALCulate[1]:LIMit:FCOut?`

Returns the total number of limit failures, in this case 2.

1 Power Sensor Remote Operation

NOTE

If `TRIGger:DElay:AUTO` is set to ON, then the number of failures returned by `CALCulate[1]:LIMit:FCOunt?` is affected by the current filter settings.

Getting the Best Speed Performance

This section discusses the factors that influence the speed of operation (number of readings/sec) of U2000 Series USB power sensors.

The following factors are those which have the greatest effect upon measurement speed (in no particular order):

- The selected measurement rate, i.e. `NORMAL`, `DOUBLE`, `FAST`.
- The trigger mode (for example, free run, trigger with delay etc.).
- The output format: `ASCII` or `REAL`.
- The units used for the measurement.
- The command used to take a measurement.

In addition, in `FAST` mode there are other influences which are described in “[Fast Mode](#)” on page 34.

The following paragraphs give a brief description of the above factors and how they are controlled from SCPI.

Measurement Rate

There are three possible speed settings `NORMAL`, `DOUBLE` and `FAST`. These are set using the `[SENSE[1]]:MRATE` command.

In `NORMAL` and `DOUBLE` modes, full instrument functionality is available. In `FAST` mode averaging, limits are disabled.

Refer to “Specifications” in the *U2000 Series USB Power Sensors Operating and Service Guide* to see the influence of these speed settings on the accuracy and noise performance of the power sensor.

Trigger Mode

The power sensor has a very flexible triggering system. For simplicity, it can be described as having three modes:

1 Power Sensor Remote Operation

- **Free Run:** When a channel is in Free Run, it continuously takes measurements. A measurement is in free run when `INITiate:CONTinuous` is set to `ON` and `TRIGger:SOURce` is set to `IMMediate`.
- **Triggered Free Run:** When a channel is in Triggered Free Run Continuous Trigger, it takes a new measurement each time a trigger event is detected. A channel is in Triggered Free Run Continuous Trigger when `INITiate:CONTinuous` is set to `ON` and `TRIGger:SOURce` is not set to `IMMediate`.
- **Single Shot:** When a channel is in Single Shot, it takes a new measurement when a trigger event is detected and then returns to the idle state. A channel is in Single Shot when `INITiate:CONTinuous` is set to `OFF`. Note that a measurement can take several `EXT` triggers depending on the filter settings. Refer to “`TRIGger[1]:DELay:AUTO <boolean>`” on page 273 for further information.

NOTE

A trigger event can be any of the following:

- The input signal meeting the trigger level criteria.
- Auto-level triggering being used.
- A `TRIGger[1][:IMMediate]` or `*TRG` command being sent.
- An external TTL level trigger being detected.

Trigger with Delay

This can be achieved using the same sequences above (apart from the second) with `TRIG:DEL:AUTO` set to `ON`. Also, the `MEAS?` command operates in trigger with delay mode.

In trigger with delay mode, a measurement is not completed until the power sensor filter is full. In this way, the reading returned is guaranteed to be settled. In all other modes, the result returned is simply the current result from the filter and may or may not be settled. This depends on the current length of the filter and the number of readings that have been taken since a change in power level.

With trigger with delay enabled, the measurement speed can be calculated roughly using the following equation:

$$\text{readings/sec} = \text{speed (as set by [SENSe[1]]:MRATe)} / \text{filter length}$$

For example, with a filter length of 4 and `SENS[1]:MRATE` set to `NORMAL`, approximately 5 readings/sec is calculated by the power sensor.

Output Format

The power sensor has two output formats for measurement results: `ASCII` and `REAL`. These formats are selected using the `FORMAt` command. When `FORMAt` is set to `REAL`, the returned result is in IEEE 754 floating-point format (note that the byte order can be changed using `FORMAt:BOReR`) plus `<LF>` as an end sentinel of the block.

The `REAL` format is likely to be required only for `FAST` mode as it reduces the amount of bus traffic.

Units

The power sensor can output results in either linear or log units. The internal units are linear, therefore optimal performance is achieved when the results output are also in linear units (since the overhead of performing a log function is removed).

Command Used

In Free Run mode, `FETCh?` must be used to return a result.

In other trigger modes, there are a number of commands which can be used, for example, `MEASure?`, `READ?`, `FETCh?` Note that the `MEAS?` and `READ?` commands are compound commands—they perform a combination of other lower level commands. Typically, the best speed performance is achieved using the low level commands directly.

Trigger Count

To get the fastest measurement speed `TRIG:COUNT` must be set to return multiple measurements for each `FETCh` command. For average only measurements a count of 4 is required, however, 10 is recommended.

Fast Mode

In the highest speed setting, the limiting factor tends to be the speed of the controller being used to retrieve results from the power sensor, and to a certain extent, the volume of remote traffic. The latter can be reduced using the `FORMat REAL` command to return results in binary format. The former is a combination of two factors:

- the hardware platform being used
- the programming environment being used

How Measurements are Calculated

Figure 1-5 details how measurements are calculated. It shows the order in which the various power sensor functions are implemented in the measurement calculation.

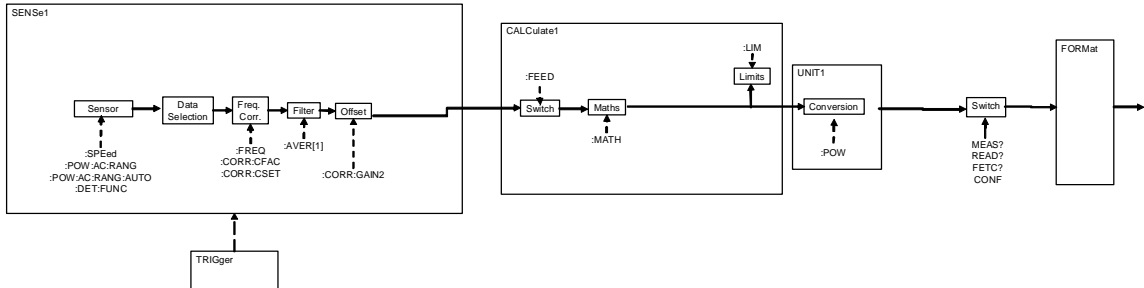


Figure 1-5 How Measurement are Calculated

The MEASure commands in this figure can be replaced with the FETCh? and READ? commands.

Status Reporting

Status reporting is used to monitor the power sensor to determine when events have occurred. Status reporting is accomplished by configuring and reading status registers.

The power sensor has the following main registers:

- Status Register
- Standard Event Register
- Operation Status Register
- Questionable Status Register
- Device Status Register

There are other registers that exist “behind” the main registers, and are described later in this chapter.

Status and Standard Event registers are read using the IEEE-488.2 common commands.

Operation and Questionable Status registers are read using the SCPI `STATus` command subsystem.

The General Status Register Model

The generalized status register model shown in [Figure 1-6](#) is the building block of the SCPI status system. This model consists of a condition register, a transition filter, an event register and an enable register. A set of these registers is called a status group.

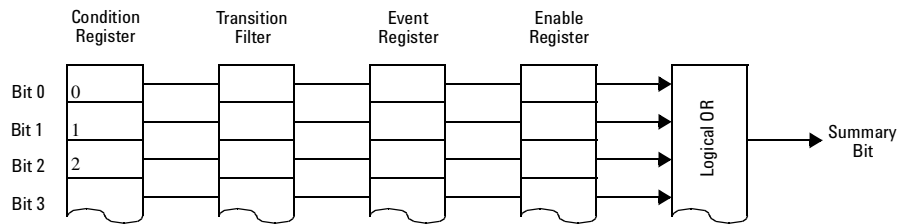


Figure 1-6 Generalized Status Register Model

When a status group is implemented in an instrument, it always contains all of the component registers. However, there is not always a corresponding command to read or write to every register.

Condition Register

The condition register continuously monitors the hardware and firmware status of the power sensor. There is no latching or buffering for this register, it is updated in real time. Condition registers are read-only.

Transition Filter

The transition filter specifies which types of bit state changes in the condition registers and set corresponding bits in the event register. Transition filter bits may be set for positive transitions (PTR), negative transitions (NTR), or both. Transition filters are read-write. They are unaffected by *CLS or queries. After `STATUS:PRESet` the NTR register is set to 0 and all bits of the PTR are set to 1.

Event Register

The event register latches transition events from the condition register as specified by the transition filter. Bits in the event register are latched and on setting they remain set until cleared by a query or a *CLS. Also on setting, an event bit is no longer affected by condition changes. It remains set until the event register is cleared; either when you read the register or when you send the *CLS (clear status) command. Event registers are read-only.

Enable Register

The enable register specifies the bits in the event register that can generate a summary bit. The instrument logically ANDs corresponding bits in the event and enable registers and ORs all the resulting bits to obtain a summary bit. Enable registers are read-write. Querying an enable register does not affect it.

An Example Sequence

Figure 1-7 illustrates the response of a single bit position in a typical status group for various settings. The changing state of the condition in question is shown at the bottom of the figure. A small binary table shows the state of the chosen bit in each status register at the selected times T1 to T5.

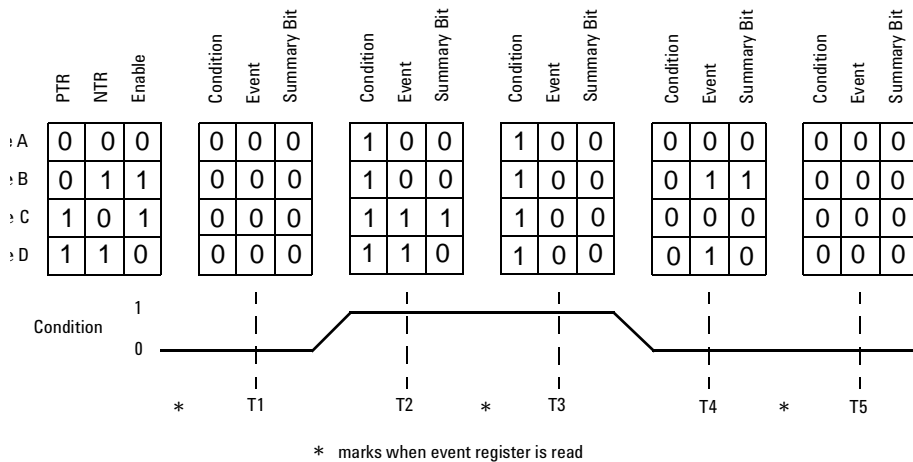


Figure 1-7 Typical Status Register Bit Changes

How to Use Register

There is a method to access the information in status groups:

- the polling method

Use the polling method when:

- your language/development environment does not support SRQ interrupts.
- you want to write a simple, single purpose program and do not want to add the complexity of setting an SRQ handler.

The Condition Polling Method

In this polling method, the power sensor has a passive role. It only informs the controller that conditions have changed when the controller asks. When you monitor a condition with the polling method, you must:

- 1 Determine which register contains the bit that monitors the condition.
- 2 Send the unique query that reads that register.
- 3 Examine the bit to see if the condition has changed.

The polling method works well if you do not need to know about the changes the moment they occur. Detecting an immediate change in a condition using the polling method requires your program to continuously read the registers at very short intervals. This is not particularly efficient and there is a possibility that an event may be missed.

Status Registers

The Status System in the power sensor is shown in [Figure 1-8](#). The Operation Status and Questionable Status groups are 16 bits wide, while the Status Byte and Standard Event groups are 8 bits wide. In all 16-bit groups, the most significant bit (bit 15) is not used and is always set to 0.

1 Power Sensor Remote Operation

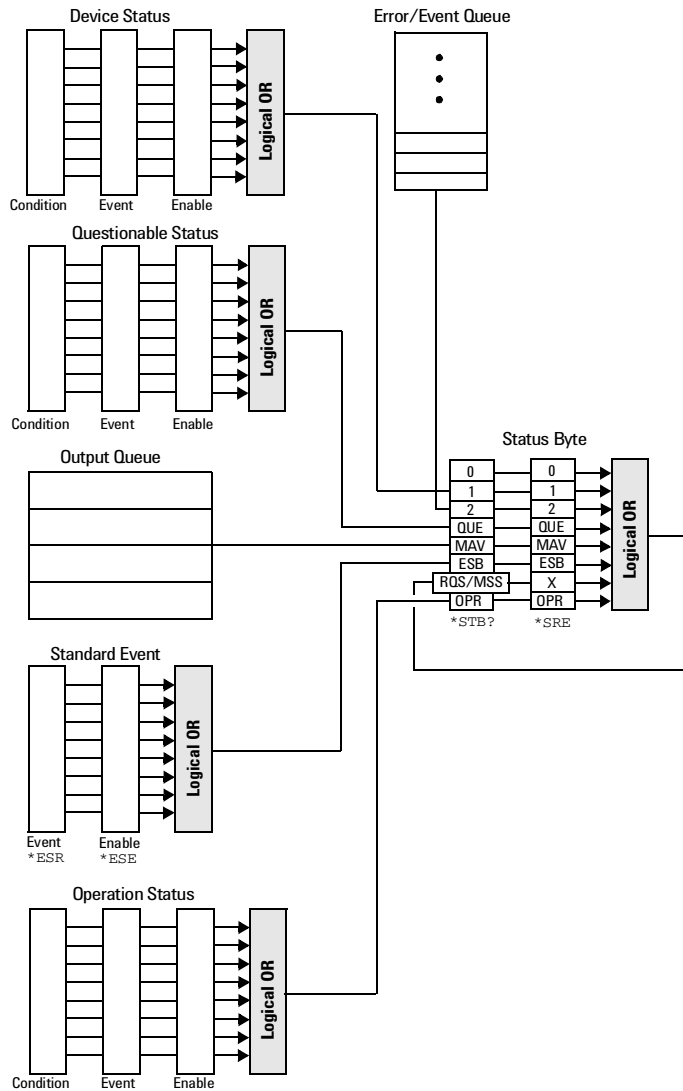


Figure 1-8 Status System

The Status Byte Summary Register

The status byte summary register reports conditions from other status registers. Query data waiting in the power sensor's output buffer is immediately reported through the "message available" bit (bit 4). Clearing an event register clears the corresponding bits in the status byte summary register. Reading all messages in the output buffer, including any pending queries, clears the message available bit.

Table 1-3 Bit Definitions - Status Byte Register

Bit Number	Decimal Weight	Definition
0	1	Not Used (Always set to 0)
1	2	Device Status Register summary bit. One or more bits are set in the Device Status Register (bits must be "enabled" in enable register)
2	4	Error/Event Queue
3	8	Questionable Status Register summary bit. One or more bits are set in the Questionable Status Register (bits must be "enabled" in enable register).
4	16	Data Available Data is available in the power sensor's output buffer.
5	32	Standard Event One or more bits are set in the Standard Event register (bits must be "enabled" in enable register).
6	64	Request Service The power sensor is requesting service (serial poll).
7	128	Operation Status Register summary bit. One or more bits are set in the Operation Status Register (bits must be "enabled" in enable register).

Particular bits in the status byte register are cleared when:

- The standard event, Questionable status, operation status and device status are queried.
- The error/event queue becomes empty.
- The output queue becomes empty.

1 Power Sensor Remote Operation

The status byte enable register (SRE, service request enable) is cleared when you:

- cycle the instrument power.
- execute a *SRE 0 command.

Using *STB? to Read the Status Byte

The *STB? (status byte query) command is similar to a serial poll except it is processed like any other power sensor command. The *STB? command returns the same result as an IEEE-488 serial poll except that the request service bit (bit 6) *is not* cleared if a serial poll has occurred. The *STB? command is not handled automatically by the IEEE-488 bus interface hardware and the command is executed only after previous commands have completed. Using the *STB? command does not clear the status byte summary register.

The Standard Event Register

The standard event register reports the following types of instrument events: power-on detected, command and syntax errors, command execution errors, self-test or calibration errors, query errors, or when an overlapped command completes following a *OPC command. Any or all of these conditions can be reported in the standard event summary bit through the enable register. You must write a decimal value using the *ESE (event status enable) command to set the enable register mask.

Table 1-4 Bit Definitions - Standard Event Register

Bit Number	Decimal Value	Definition
0	1	Operation Complete All overlapped commands following an *OPC command have been completed.
1	2	Not Used. (Always set to 0.)
2	4	Query Error A query error occurred, refer to error numbers 410 to 440 in the "Error Message List" in Chapter SYSTem Subsystem.

Bit Number	Decimal Value	Definition
3	8	Device Error A device error occurred, refer to error numbers 310 to 350 in the "Error Message List" in Chapter SYSTem Subsystem.
4	16	Execution Error An execution error occurred, refer to error numbers 211 to 231 in the "Error Message List" in Chapter SYSTem Subsystem.
5	32	Command Error A command syntax error occurred, refer to error numbers 101 to 161 in the "Error Message List" in Chapter SYSTem Subsystem.
6	64	User request.
7	128	Power On Power has been turned off and on since the last time the event register was read or cleared.

The standard event register is cleared when you:

- send a *CLS (clear status) command.
- query the event register using the *ESR? (event status register) command.

The standard event enable register is cleared when you:

- cycle the instrument power.
- execute a *ESE 0 command.

Questionable Status Register

The questionable status register provides information about the quality of the power sensor's measurement results. Any or all of these conditions can be reported in the questionable data summary bit through the enable register. You must write a value using the `STATUS:QUESTIONABLE:ENABLE` command to set the enable register mask.

The questionable status model is shown in the pullout at the end of this chapter.

The following bits in these registers are used by the power sensor.

Table 1-5 Bit Definitions - Questionable Status Registers

Bit Number	Decimal Weight	Definition
0 to 2	-	Not used
3	8	POWer Summary
4 to 7	-	Not used
8	256	CALibration Summary
9	512	Power On Self Test
10 to 14	-	Not Used
15	-	Not used (always 0)

The condition bits are set and cleared under the following conditions:

Table 1-6 Bit change conditions for Questionable Status Register

Bit Number	Meaning	EVENTs Causing Bit Changes
3	POWer Summary	<p>This is a summary bit for the Questionable POWer Register.</p> <ul style="list-style-type: none"> • SET: Error –230, “Data corrupt or stale” • CLEARED: When no errors are detected by the power sensor during a measurement covering the causes given for it to set.

Bit Number	Meaning	EVENTs Causing Bit Changes
8	CALibration Summary	<p>This is a summary bit for the Questionable CALibration Register.</p> <ul style="list-style-type: none"> SET: These may be caused by CALibration[1]:ZERO:AUTO ONCE or CALibration[1]:AUTO ONCE or CALibration[1][:ALL] or CALibration[1][:ALL]?. Error -231, "Data questionable; ZERO ERROR" Error -231, "Data questionable; CAL ERROR" CLEARED: When any of the commands listed above succeed and no errors are placed on the error queue.
9	Power On Self Test	<ul style="list-style-type: none"> SET: This bit is set when the power on self test fails. CLEARED: When the power on self test passes.

Operation Status

The Operation Status group monitors conditions in the power sensor's measurement process.

The Operation status model is shown in the pullout at the end of this chapter.

The following bits in these registers are used by the power sensor:

Table 1-7 Bit Definitions - Operation Status

Bit Number	Decimal Weight	Definition
0	1	CALibrating Summary
1 - 3	-	Not used
4	16	MEASuring Summary
5	32	Waiting for TRIGger Summary
6 - 9	-	Not used
10	1024	SENSe Summary

1 Power Sensor Remote Operation

Bit Number	Decimal Weight	Definition
11	2048	Lower Limit Fail Summary
12	4096	Upper Limit Fail Summary
13 to 14	-	Not used
15	-	Not used (always 0)

The condition bits are set and cleared under the following conditions:

Table 1-8 Bit change conditions for Operation Status

Bit Number	Meaning	EVENTs Causing Bit Changes
0	CALibrating	<p>This is a summary bit for the Operation CALibrating Register.</p> <ul style="list-style-type: none"> SET: At beginning of zeroing (CALibration:ZERO:AUTO ONCE) and at the beginning of calibration (CALibration:AUTO ONCE). Also for the compound command/query CALibration[:ALL]?, this bit is set when sensor zeroing begins. CLEARED: At the end of zeroing or calibration.
4	MEASuring	<p>This is a summary bit for the Operation MEASuring Register.</p> <ul style="list-style-type: none"> SET: When the power sensor is taking a measurement. CLEARED: When the measurement is finished.
5	Waiting for TRIGger	<p>This is a summary bit for the Operation TRIGger Register.</p> <ul style="list-style-type: none"> SET: When the power sensor enters the “wait for trigger” state. CLEARED: When the power sensor enters the “idle” state.
10	SENSe	<p>This is a summary bit for the Operation SENSe Register.</p> <ul style="list-style-type: none"> SET: When the power sensor is reading data from non-volatile memory. CLEARED: When the power sensor is not reading data from non-volatile memory.
11	Lower Limit Fail	<p>This is a summary bit for the Lower Limit Fail Register.</p> <ul style="list-style-type: none"> SET: If a measurement is made and lower limit test fails. CLEARED: If a measurement is made and the lower limit test is not enabled or the test is enabled and passes.

Bit Number	Meaning	EVENTs Causing Bit Changes
12	Upper Limit Fail	<p>This is a summary bit for the Upper Limit Fail Register.</p> <ul style="list-style-type: none"> • SET: If a measurement is made and upper limit test fails. • CLEARED: If a measurement is made and the upper limit test is not enabled or the test is enabled and passes.

Device Status Register

The device status register set contains bits which give device dependent information.

The following bits in these registers are used by the power sensor:

Table 1-9 Bit Definitions - Device Status Register

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Not used
2	4	Not used
3	8	Power sensor error
4	16	Not used
5	32	Not used
6	64	Not used
14	16384	Not used

The condition bits are set and cleared under the following conditions:

1 Power Sensor Remote Operation

Table 1-10 Bit change conditions for Device Status Register

Bit Number	Meaning	EVENTs Causing Bit Changes
3	Power sensor error	<ul style="list-style-type: none">• SET: If the power sensor non-volatile memory has failed or other hardwares have failed.• CLEARED: In every other condition.

Using the Operation Complete Commands

The *OPC? and *OPC commands allow you to maintain synchronization between the computer and the power sensor. The *OPC? query command places an ASCII character 1 into the power sensor's output queue when all pending power sensor commands are complete. If your program reads this response before continuing program execution, you can ensure synchronization between one or more instruments and the computer.

The *OPC command sets bit 0 (Operation Complete) in the Standard Event Status Register when all pending power sensor operations are complete.

Procedure

- Send a device clear message to clear the power sensor's output buffer.
- Clear the event registers with the *CLS (clear status) command.
- Enable operation complete using the *ESE 1 command (standard event register).
- Send the *OPC? (operation complete query) command and enter the result to assure synchronization.
- Send your programming command string, and place the *OPC (operation complete) command as the last command.
- Send the *STB? (status byte query) command to poll the register. This command does not clear the status byte summary register.

Examples

This example program uses the *OPC? command to determine when the power sensor has finished calibrating.

```
CAL:AUTO ONCE
*OPC?
MEAS:POW:AC?
```

Saving and Recalling Power Sensor Configurations

To reduce repeated programming, up to ten power sensor configurations can be stored in the non-volatile memory. The error list, zeroing and calibration information are not stored.

How to Save and Recall a Configuration

Power sensor configurations are saved and recalled with the following commands:

*SAV <NRf>

*RCL <NRf>

The range of values for <NRf> in the above commands is 1 to 10.

Using Device Clear to Halt Measurements

Device clear is an IEEE-488 low-level bus message which can be used to halt measurements in progress. Different programming languages and IEEE-488 interface cards provide access to this capability through their own unique commands. 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.

- All measurements in progress are aborted.
- The power sensor returns to the trigger “idle state”.
- The power sensor’s input and output buffers are cleared.
- The power sensor is prepared to accept a new command string.

NOTE

For interfaces that do not support a low-level device clear, use the `ABORT` command.

An Introduction to the SCPI Language

Standard Commands for Programmable Instruments (SCPI) defines how you communicate with an instrument from a bus controller. The SCPI language uses a hierarchical structure similar to the file systems used by many bus controllers. The command tree is organized with root-level commands (also called subsystems) positioned at the top, with multiple levels below each root-level command. You must specify the complete path to execute the individual lower-level commands.

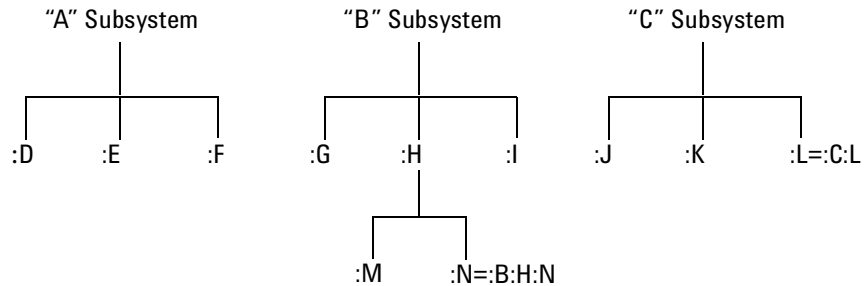


Figure 1-9 Hierarchical structure of SCPI

Mnemonic Forms

Each keyword has both a long and a short form. A standard notation is used to differentiate the short form keyword from the long form keyword. The long form of the keyword is shown, with the short form portion shown in uppercase characters, and the rest of the keyword shown in lowercase characters. For example, the short form of TRIGger is TRIG.

Using a Colon (:)

When a colon is the first character of a command keyword, it indicates that the next command mnemonic is a root-level command. When a colon is inserted between two command mnemonics, the colon moves the path

down one level in the present path (for the specified root-level command) of the command tree. You *must* separate command mnemonics from each other using a colon. *You can omit the leading colon if the command is the first of a new program line.*

Using a Semicolon (;)

Use a semicolon to separate two commands within the same command string. The semicolon does not change the present path specified. For example, the following two statements are equivalent. Note that in the first statement the first colon is optional but the third is compulsory.

```
CALibration[1]:ZERO:TYPE EXtErnal;CALibration[1]:ZERO:AUTO ONCE
CALibration[1]:ZERO:TYPE EXtErnal;ZERO:AUTO ONCE
```

Using a Comma (,)

If a command requires more than one parameter, you must separate adjacent parameters using a comma.

Using Whitespace

You *must* use whitespace characters, [tab], or [space] to separate a parameter from a command keyword. Whitespace characters are generally ignored *only* in parameter lists.

Using “?” Commands

The bus controller may send commands at any time, but a SCPI instrument may only send responses when *specifically* instructed to do so. Only query commands (commands that end with a “?”) instruct the instrument to send a response message. Queries return either measured values or internal instrument settings.

NOTE

If you send two query commands without reading the response from the first, then attempt to read the second response, you may receive some data from the first response followed by the complete second response. To avoid this, do not send a query command without reading the response. When you cannot avoid this situation, send a device clear before sending the second query command.

Using "*" Commands

Commands starting with a "*" are called common commands. They are required to perform the identical function for *all* instruments that are compliant with the IEEE-488.2 interface standard. The "*" commands are used to control reset, self-test, and status operations in the power sensor.

Syntax Conventions

Throughout this guide, the following conventions are used for SCPI command syntax.

- Square brackets ([]) indicate optional keywords or parameters.
- Braces ({}) enclose one or more parameters that may be included zero or more times.
- Triangle brackets (<>) indicate that you must substitute a value for the enclosed parameter.
- Bars (|) can be read as "or" and are used to separate alternative parameter options.

Syntax Diagram Conventions

- Solid lines represent the recommended path.
- Ovals enclose command mnemonics. The command mnemonic must be entered exactly as shown.

- Dotted lines indicate an optional path for by passing secondary keywords.
- Arrows and curved intersections indicate command path direction.

SCPI Data Types

The SCPI language defines different data formats for use in program messages and response messages. Instruments are flexible listeners and can accept commands and parameters in various formats. However, SCPI instruments are precise talkers. This means that SCPI instruments *always* respond to a particular query in a predefined, rigid format.

<boolean> Definition

Throughout this document <boolean> is used to represent ON|OFF|<NRf>. boolean parameters have a value of 0 or 1 and are unitless. ON corresponds to 1 and OFF corresponds to 0.

On input, an <NRf> is rounded to an integer. A nonzero result is interpreted as 1.

Queries always return a 1 or 0, never ON or OFF.

<character_data> Definition

Throughout this document <character_data> is used to represent character data, that is, A -Z, a -z, 0 -9 and _ (underscore). For example: START and R6_5F. The format is defined as:

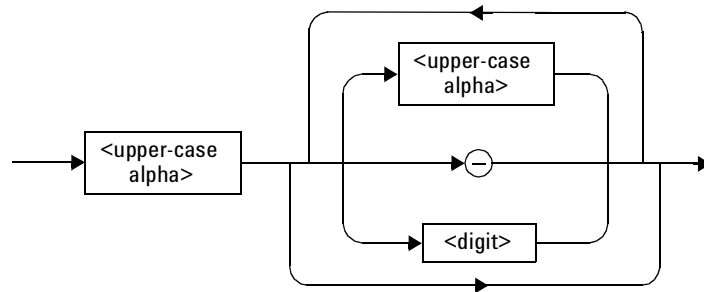


Figure 1-10 Format of <character_data>

<NAN> Definition

Not a number (NAN) is represented as 9.91 E37. Not a number is defined in IEEE 754.

<non-decimal numeric> Definition

Throughout this document <non-decimal numeric> is used to represent numeric information in bases other than ten (that is, hexadecimal, octal and binary). The following syntax diagram shows the standard for these three data structures. For examples, #HA2F, #ha4e, #Q62, #q15, #B01011.

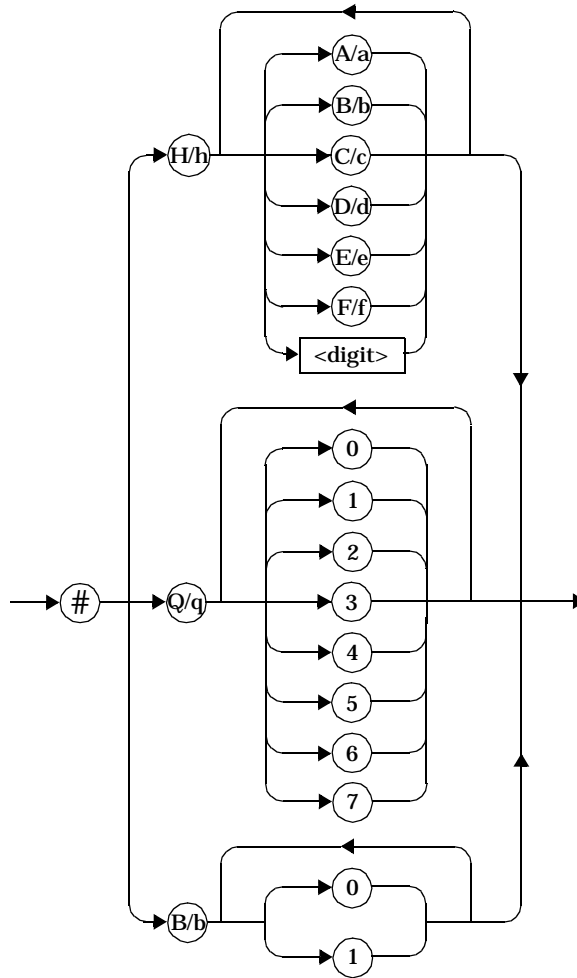


Figure 1-11 Format of <non-decimal numeric>

Refer to section 7.7.4.1 of IEEE 488.2 for further details.

<NRf> Definition

Throughout this document <NRf> is used to denote a flexible numeric representation. For example: +200; -56; +9.9E36. Refer to section 7.7.2.1 of

1 Power Sensor Remote Operation

IEEE 488.2 for further details.

<NR1> Definition

Throughout this document <NR1> numeric response data is defined as:

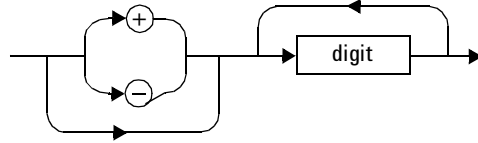


Figure 1-12 Format of <NR1>

For example:

- 146
- +146
- -12345

Refer to section 8.7.2 of IEEE 488.2 for further details.

<NR2> Definition

Throughout this document <NR2> numeric response data is defined as:

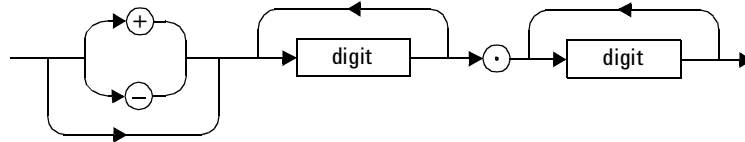


Figure 1-13 Format of <NR2>

For example:

- 12.3
- +1.2345
- -0.123

Refer to section 8.7.3 of IEEE 488.2 for further details.

<NR3> Definition

Throughout this document <NR3> numeric response data is defined as:

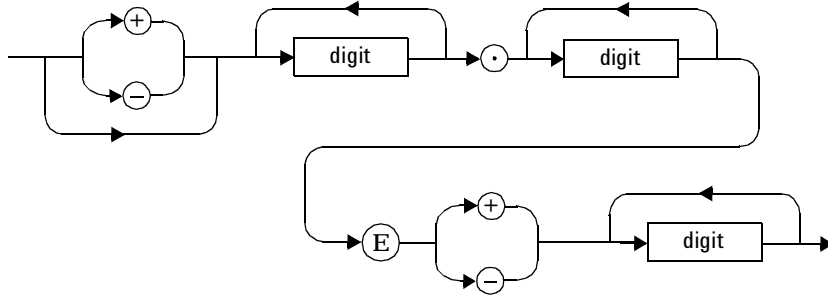


Figure 1-14 Format of <NR3>

For example:

- 1.23E+6
- 123.4E-54
- -1234.567E+90

Refer to section 8.7.4 of IEEE 488.2 for further details.

<numeric_value> Definition

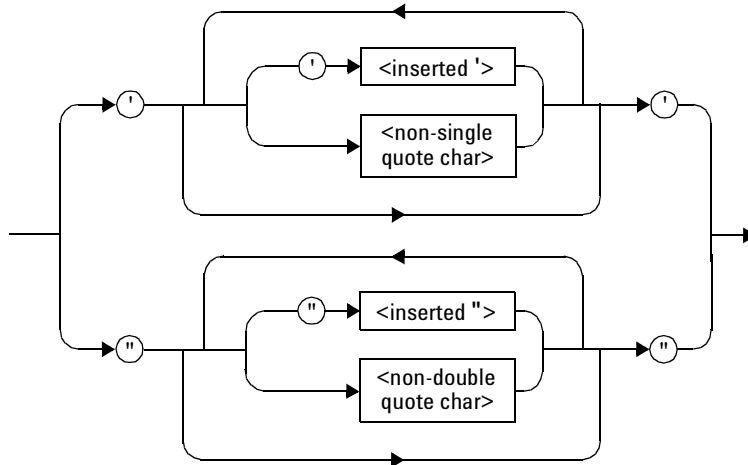
Throughout this document the decimal numeric element is abbreviated to <numeric_value>. For example, <NRf>, MINimum, MAXimum, DEFault or Not A Number (NaN).

<string> Definition

Throughout this document <string> is used to represent 7-bit ASCII characters.

The format is defined as:

Program Data



Response Data

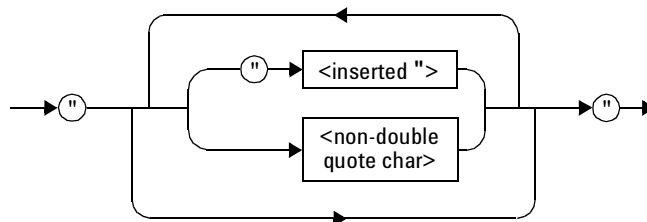


Figure 1-15 Format of <string>

Input Message Terminators

Program messages sent to a SCPI instrument *must* terminate with a <newline> character. The IEEE.488 EOI (end or identify) signal is interpreted as a <newline> character and may also be used to terminate a message in place of the <newline> character. A <carriage return> followed by a <newline> is also accepted. Many programming languages allow you to specify a message terminator character or EOI state to be automatically sent with each bus transaction. Message termination *always* sets the current path back to the root-level.

SCPI Compliance Information

The power sensor complies with the rules and regulations of the present version of SCPI (Standard Commands for Programmable Instruments). You can determine the SCPI version with which the power sensor's is in compliance by sending the `SYSTem:VERSion?` command from the remote interface.

The following commands are device-specific to the power sensor. They are not included in the 1999.0 version of the SCPI standard. However, these commands are designed with the SCPI format in mind and they follow all of the syntax rules of the standard.

```
MEMory:CLEar[:NAME]
MEMory:TABLE:SElect
MEMory:STATe:DEFine
MEMory:TABLE:GAIN[:MAGNitude]
MEMory:TABLE:GAIN:POINTs?
MEMory:TABLE:MOVE
[SENSe[1]]:AVERage:SDETect
[SENSe[1]]:CORRection:FDOFfset
[SENSe[1]]:MRATe
[SENSe[1]]:POWER:AC:RANGe
SERVice:SENSor[1]:CDATe?
SERVice:SENSor[1]:CPLace?
SERVice:SENSor[1]:SNUMber?
SERVice:SENSor[1]:TYPE?
```


Summary of Commands

For detail of each SCPI (Standard Commands for Programmable Instruments) command available to program the power sensor, refer to later chapters for more details on each command.

All the commands listed also have queries unless otherwise stated in the “Notes” column.

1 Power Sensor Remote Operation



2 MEASurement Commands

MEASurement Commands	66
CONFigure[1]?	69
CONFigure [1] Commands	71
CONFigure[1][:SCALar][:POWer:AC] [<expected_value>[,<resolution>[,<source list>]]]	72
FETCh[1]?	74
FETCh[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	75
READ[1] Commands	77
READ[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	78
MEASure[1] Commands	80
MEASure[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]	81

This chapter explains how to use the MEASure group of instructions to acquire data using a set of high level instructions.



MEASurement Commands

Measurement commands are high level commands used to acquire data. They enable you to trade interchangeability against fine control of the measurement process.

Measurement Command	Descriptions
MEASure?	Provides the simplest way to program a power sensor for measurements. MEASure? is a compound command which is equivalent to an ABORT followed by a CONFigure and a READ?. It does not enable much flexibility or control over measurement settings.
CONFigure	Used to change the power sensor's configuration values. CONFigure must then be followed by another command which takes the measurement—for example, a READ? followed by a FETCh?.
READ?	Takes a measurement using parameters previously set up using either CONFigure or lower level commands. READ? is equivalent to an ABORT followed by an INITiate1 (which performs the data acquisition) and a FETCh?
FETCh?	Retrieves measurements taken by INITiate*.

* INITiate is described in [Chapter 11, "TRIGger Subsystem,"](#) on page 261.

The CONFigure, FETCh?, READ? and MEASure? commands all have a numeric suffix which refers to a specific window/measurement. [Figure 2-1](#) shown an example of the configuration returned result windows.



Figure 2-1 Measurement Display CALCulate Block Window

Optional Parameters

CONFigure, FETCh?, READ? and MEASure? have the following three optional parameters:

- An expected power value
- A resolution
- A source list

Expected Power Value

An `<expected_value>` parameter is only required if you are using an U2000 Series USB Power Sensor. The value entered determines which of the power sensor's two ranges is used for the measurement. If the current setting of the power sensor's range is no longer valid for the new measurement, specifying the expected power value decreases the time taken to obtain a result.

Resolution

The `<resolution>` parameter sets the resolution of the current window. This parameter does not affect the resolution of the remote data but it does affect the auto averaging setting. The highest resolution setting is taken to calculate the averaging. If you are making a ratio or difference measurement the `<resolution>` parameters are applied to both.

Source List

The `<source list>` parameter is used to define:

2 MEASurement Commands

- What channels the measurements will be made on, for a dual measurement.
- Whether the calculation is A-B or B-A, for a two difference measurement.
- Whether the calculation is A/B or B/A, for a ratio measurement.

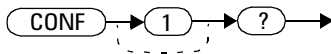
The following commands are described in this chapter:

Keyword	Parameter Form	Notes	Page
CONFigure[1]		[query only]	page 69
CONFigure[1] [:SCALar] [:POWER:AC]	[<expected_value> [,<resolution>[,<source list>]]]	[no query]	page 72
FETCh[1] [:SCALar] [:POWER:AC]?	[<expected_value> [,<resolution>[,<source list>]]]	[query only]	page 75
READ[1] [:SCALar] [:POWER:AC]?	[<expected_value> [,<resolution>[,<source list>]]]	[query only]	page 78
MEASure[1] [:SCALar] [:POWER:AC]?	[<expected_value> [,<resolution>[,<source list>]]]	[query only]	page 81

CONFigure[1]?

This query returns the present configuration of the current window/measurement.

Syntax



The string returned depends on the setting of the CALCulate:MATH commands.

The configuration is returned as a quoted string in the following format:

"<function> <expected_value>,<resolution>,<source list>"

<expected_value> returns the expected value sent by the last CONFigure command or +20 dBm by default.

Example

CONF1?

This command queries the configuration of the current window/measurement.

Reset Condition

On reset:

- The command function is set to :POWER:AC.
- The expected power level is set to +20 dBm.
- The resolution is set to 3.
- The source list on the U2000 Series USB power sensors is set to (@1).

CONFigure [1] Commands

The CONFigure commands are used on the current window/measurement to set:

- The expected power level being measured.
- The resolution of the window/measurement.

The CONFigure commands do not make the power measurement after setting the configuration. Use READ?, or alternatively use INITiate followed by a FETCh? to make the measurement.

The CONFigure command also applies the following defaults to the measurement(s) which are in the current window (the measurement(s) in the window are specified in the <source list> parameter):

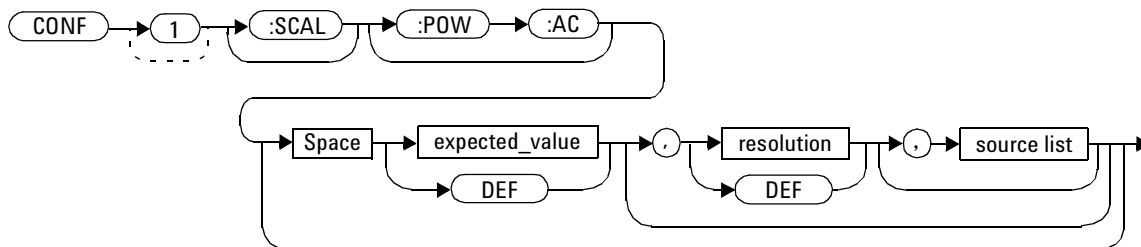
Default Settings	Description
INITiate:CONTinuous OFF	Sets the power sensor to make one trigger cycle when INITiate is sent.
TRIGger:SOURce IMMEDIATE	When TRIG:SOUR is set to BUS or HOLD, sets the power sensor to make the measurement immediately a trigger is received.
TRIGger:DELay:AUTO ON	Enables automatic delay before making the measurement.
[SENSe[1]]:AVERage:COUNT:AUTO ON	Enables automatic filter length selection.
[SENSe[1]]:AVERage:STATe ON	Enables averaging.

CONFigure[1][:SCALar][:POWer:AC] [<expected_value>[,<resolution>[,<source list>]]]

This command is used on the current window/measurement to set:

- The expected power level of the measurement.
- The resolution of the window/measurement..

Syntax



Parameters

Refer to “[Optional Parameters](#)” on page 67 for additional details on the parameters in this command.

Item	Description/Default	Range of Values
expected_value	A numeric value for the expected power level. The units of measurement are dBm and W. The default units are defined by UNIT:POWer.	Sensor dependent. DEF ¹
resolution	A numeric value for the resolution. If unspecified the current resolution setting is used.	1 to 4 ² 1.0, 0.1, 0.01, 0.001 DEF ¹

Item	Description/Default	Range of Values
source list	The measurement which the command is implemented on. If unspecified the current window setup is used.	(@1)

¹ The mnemonic DEF means DEFault. This is not equivalent to the DEFault parameter used in the command sub-systems. The parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder. Specifying DEF leaves the parameter value unchanged.

² When the measurement result is linear this parameter represents the number of significant digits. When the measurement result is logarithmic 1 to 4 represents of 1, 0.1, 0.01 and 0.001 respectively.

Example

```
CONF1:POW:AC DEF,2,(@1)
```

This command configures the current window/measurement to measure the power of the sensor, using the current sensor range and a resolution setting of 2.

FETCh[1]?

The FETCh? queries set the current window's measurement function. This can be only set to single measurement. They then recalculate the measurement and place the result on the bus. The format of the result is set by FORM[:READ][:DATA]. Refer to [Chapter 5](#), "FORMat Subsystem," on page 119 for further information.

The query returns a measurement result when it is valid. The measurement result is invalid under the following conditions:

- When *RST is executed.
- Whenever a measurement is initiated.
- When any SENSE parameter, such as frequency, is changed.

If data is invalid, the FETCh? query is not completed until all data becomes valid. The exceptions to this are, if the power sensor is in the idle state and the data is invalid, or the power sensor has been reconfigured as defined above and no new measurement has been initiated. In such cases, the FETCh? routine generates the error -230, "Data corrupt or stale" and no result is returned. A common cause for this error is receiving a FETCh? after a *RST. If the expected value and resolution parameters are not the same as those that were used to collect the data, error -221, "Settings conflict" occurs.

NOTE

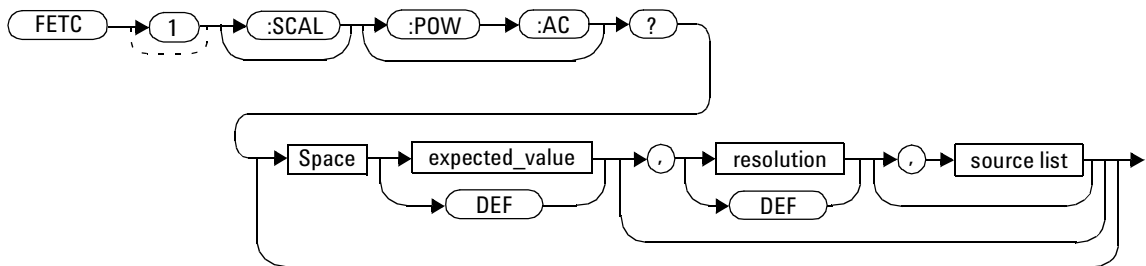
When TRIG:SOUR is EXT and a new acquisition has been initiated (using the INIT command for example), FETCh? waits until the trigger takes place before executing. If trigger conditions are not satisfied - when the trigger level differs greatly from the signal level for example - this can give the impression that the power sensor has hung.

To unlock the power sensor and adjust trigger settings, an execute clear (*CLS) must be performed.

FETCh[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]

This command sets the current window's measurement function with relative mode off, recalculates the measurement and places the result on the bus. The result is a power based measurement and is expressed in the units defined by `UNIT[1]:POWer`.

Syntax



Parameters

Refer to “[Optional Parameters](#)” on page 67 for additional details on the parameters in this command.

Item	Description/Default	Range of Values
expected_value (for the expected power level)	The expected power level parameter can be set to DEF or a numeric value. If a value is entered it should correspond to that set by <code>CONFigure</code> otherwise an error occurs. The units of measurement are dBm and W. The default units are defined by <code>UNIT:POWer</code> .	-60 dBm to +20 dBm DEF ¹

Item	Description/Default	Range of Values
resolution	A numeric value for the resolution. If it is unspecified the current resolution setting is used. If a value is entered it should correspond to the current resolution setting otherwise an error occurs.	1 to 4 ² 1.0, 0.1, 0.01, 0.001 DEF ¹
source list	The measurement which the command is implemented on. If unspecified the current window setup is used.	(@1)

¹ The mnemonic DEF means DEFault. This is not equivalent to the DEFault parameter used in the command sub-systems. The parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder. Specifying DEF leaves the parameter value unchanged.

² When the measurement result is linear this parameter represents the number of significant digits. When the measurement result is logarithmic 1 to 4 represents of 1, 0.1, 0.01 and 0.001 respectively.

Example

```
FETC1:POW:AC?
```

This command queries the current window/measurement result.

Error Messages

- If the last measurement is not valid error -230, “Data corrupt or stale” occurs. A measurement is valid after it has been initiated. It becomes invalid when either a reset occurs or any measurement parameter, for example frequency, is changed.
- If the expected_value and resolution parameters are not the same as the current expected value and resolution setting on the current window, error -221, “Settings conflict” occurs.

READ[1] Commands

The READ? commands are most commonly used with the CONFIGure command to cause a new power measurement to be taken and the result returned to the output buffer. The format of the result is set by FORM[:READ][:DATA]. Refer to [Chapter 5](#), “FORMat Subsystem,” on page 119 for further information.

- The READ? query is equivalent to:

```
ABORt  
INITiate  
FETCh?
```

or

```
ABORt1  
INITiate1  
FETCh1?
```

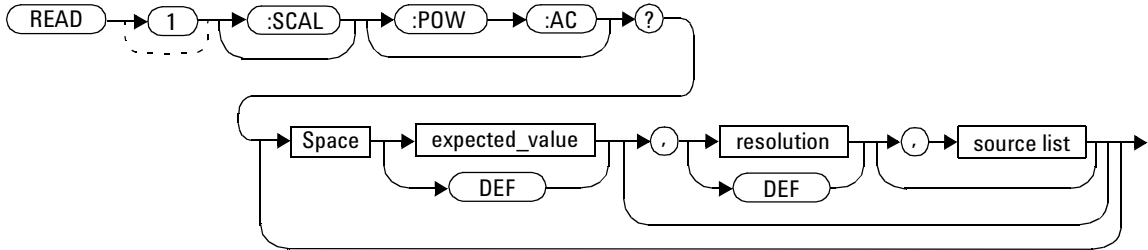
READ[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]

This command sets the current window’s measurement function with relative mode off, aborts then initiates the current measurement, calculates the measurement result and places the result on the bus. The result is a power based measurement and is expressed in the units defined by UNIT[1]:POWER.

NOTE

INITiate:CONTInuous must be set to OFF, otherwise error –213, “INIT ignored” occurs. If TRIGger:SOURce is set to BUS, error –214, “Trigger deadlock” occurs.

Syntax



Parameters

Refer to “Optional Parameters” on page 67 for additional details on the parameters in this command.

Item	Description/Default	Range of Values
expected_value (for the expected power level)	The expected power level parameter can be set to DEF or a numeric value. If a value is entered it should correspond to that set by CONFIGure otherwise an error occurs.	–60 dBm to +20 dBm DEF ¹

Item	Description/Default	Range of Values
resolution	A numeric value for the resolution. If it is unspecified the current resolution setting is used. If a value is entered it should correspond to the current resolution setting otherwise an error occurs.	1 to 4 ² 1.0, 0.1, 0.01, 0.001 DEF ¹
source list	The measurement which the command is implemented on. If unspecified the current window setup is used.	(@1)

¹ The mnemonic DEF means DEFault. This is not equivalent to the DEFault parameter used in the command sub-systems. The parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder. Specifying DEF leaves the parameter value unchanged.

² When the measurement result is linear this parameter represents the number of significant digits. When the measurement result is logarithmic 1 to 4 represents of 1, 0.1, 0.01 and 0.001 respectively.

Example

READ1:POW:AC?

This command queries the current window/measurement.

Error Messages

- INITiate:CONTinuous must be set to OFF, otherwise error -213, “INIT ignored” occurs.
- If TRIGger:SOURce is set to BUS or HOLD, error -214, “Trigger deadlock” occurs.
- If the expected value and resolution parameters are not the same as the current expected value and resolution settings on the current window, error -221, “Settings conflict” occurs.

MEASure[1] Commands

The MEASure? commands configure the power sensor to perform a power measurement with the given measurement function, range and resolution then makes the measurement. The format of the result is set by FORM[:READ][:DATA]. Refer to [Chapter 5](#), “FORMat Subsystem,” on page 119 for further information.

MEASure? is a compound command which is equivalent to:

- The MEASure? query is equivalent to:

```
ABORt  
CONFigure  
READ?
```

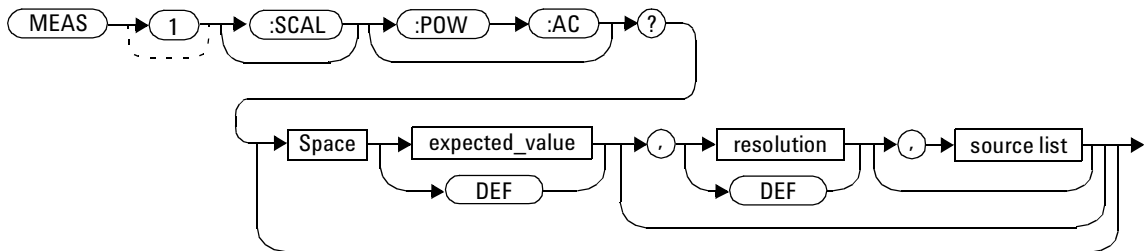
or

```
ABORt1  
CONFigure1  
READ1?
```

MEASure[1][:SCALar][:POWer:AC]? [<expected_value>[,<resolution>[,<source list>]]]

This command sets the current window's measurement function with relative mode off, aborts, configures the window, calculates the measurement result and places the result on the bus.

Syntax



Parameters

Refer to “[Optional Parameters](#)” on page 67 for additional details on the parameters in this command.

Item	Description/Default	Range of Values
expected_value (for the expected power level)	A numeric value for the expected power level. The units of measurement are dBm and W. The default units are defined by UNIT:POWer.	-60 dBm to +20 dBm DEF ¹
resolution	A numeric value for the resolution. If unspecified the current resolution setting is used.	1 to 4 ² 1.0, 0.1, 0.01, 0.001 DEF ¹

2 MEASurement Commands

Item	Description/Default	Range of Values
source list	The measurement which the command is implemented on. If unspecified the current window setup is used.	(@1)

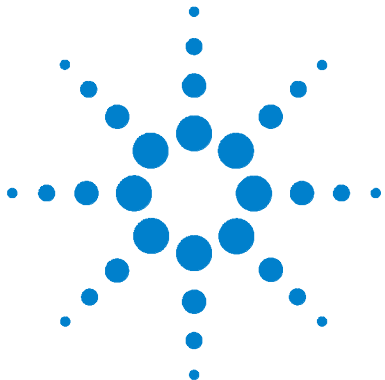
¹ The mnemonic DEF means DEFault. This is not equivalent to the DEFault parameter used in the command sub-systems. The parameters must be entered in the specified order. If parameters are omitted, they default from the right. The parameter DEFault is used as a place holder. Specifying DEF leaves the parameter value unchanged.

² When the measurement result is linear this parameter represents the number of significant digits. When the measurement result is logarithmic 1 to 4 represents of 1, 0.1, 0.01 and 0.001 respectively.

Example

```
MEAS1:POW:AC?  
-70DBM,1,(@1)
```

This command queries the current window/measurement of the sensor, using an expected power level of - 70 dBm and a resolution setting of 1.



3 CALCulate Subsystem

CALCulate Subsystem	84
CALCulate[1]:FEED[1]<string>	86
CALCulate[1]:LIMit Commands	89
CALCulate[1]:LIMit:CLEar:AUTo <boolean> ONCE	90
CALCulate[1]:LIMit:CLEar[:IMMEDIATE]	92
CALCulate[1]:LIMit:FAIL?	93
CALCulate[1]:LIMit:FCOunt?	94
CALCulate[1]:LIMit:LOWer[:DATA] <numeric_value>	96
CALCulate[1]:LIMit:UPPer[:DATA] <numeric_value>	98
CALCulate[1]:LIMit:STATe <boolean>	101
CALCulate[1]:MATH Commands	103
CALCulate[1]:MATH[:EXPRession] <string>	104
CALCulate[1]:MATH[:EXPRession]:CATalog?	106

This chapter explains how the CALCulate subsystem is used to perform post acquisition data processing.



CALCulate Subsystem

The CALCulate subsystem performs post acquisition data processing. Functions in the SENSE subsystem are related to data acquisition, while the CALCulate subsystem operates on the data acquired by a SENSE function.

There is an independent CALCulate block in the power sensor, as shown in [Figure 3-1](#).

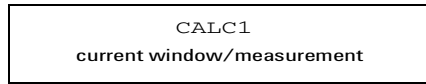


Figure 3-1 Measurement Display CALCulate Block Window

[Figure 3-2](#) details where the commands are applied within the CALCulate block.

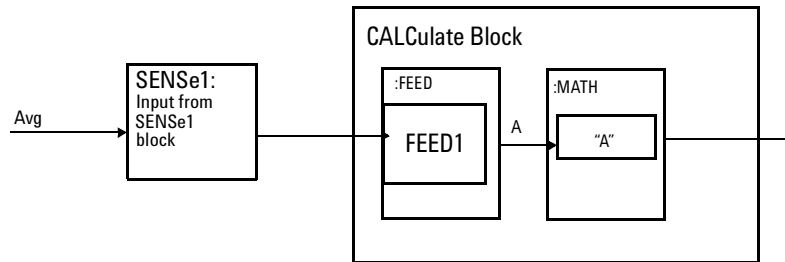


Figure 3-2 CALCulate Block

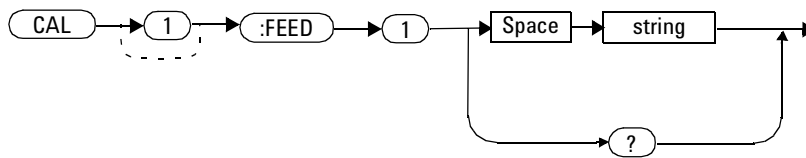
Keyword	Parameter Form	Notes	Page
CALCulate[1]			
:FEED[1]	<string>		page 86
:LIMit			
:CLEar			
:AUTO	<boolean> ONCE		page 90
[:IMMediate]			page 92
:FAIL?		[query only]	page 93
:FCOunt?		[query only]	page 94
:LOWer			
[:DATA]	<numeric_value>		page 96
:UPPer			
[:DATA]	<numeric_value>		page 98
:STATe	<boolean>		page 101
:MATH			
[:EXPRession]	<string>		page 104
:CATalog?		[query only]	page 106

CALCulate[1]:FEED[1]<string>

This command sets the input measurement mode to be fed to the specified input on the CALC block. It is applied to the measurement after the CALC:MATH:EXPR command has been used to specify which measurement the feed is taken from.

Under certain circumstances the measurement mode is changed by the CALC:MATH:EXPR command. Refer to “CALCulate[1]:MATH[:EXPRession] <string>” on page 104 for further information.

Syntax



Parameters

Item	Description	Range of Values
string	The input measurement type to be fed to the specific input on the CALC block: <ul style="list-style-type: none"> • AVER: average 	"POW:AVER"

Example

```
CALC1:FEED1 "POW:AVER"
```

This command selects the input for FEED1 of CALC block CALC1 to be average power, using gate 1. The measurement from which the feed is taken is determined by CALC:MATH:EXPR.

Reset Condition

On reset, the feed is set to :POW:AVER.

Query

```
CALCulate[1]:FEED[1]?
```

The query returns the current value of the string.

Query Example

```
CALC1:FEED1?
```

This command queries the current setting of the CALC block on FEED1 of the current window/measurement.

CALCulate[1]:LIMit Commands

These commands set the limits on the current window/measurement enabling you to:

- Set upper and lower level limits
- Query if there has been a failure
- Count the number of failures
- Clear the counter

The following commands are detailed in this section:

```
CALCulate[1]:LIMit:CLEar:AUTO <boolean>
```

```
CALCulate[1]:LIMit:CLEar[IMMediate]
```

```
CALCulate[1]:LIMit:FAIL?
```

```
CALCulate[1]:LIMit:FCOunt?
```

```
CALCulate[1]:LIMit:LOWer[:DATA]
```

```
CALCulate[1]:LIMit:UPPer[:DATA]
```

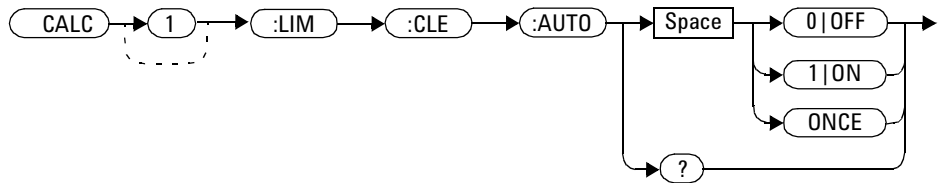
```
CALCulate[1]:LIMit:STATe <boolean>
```

CALCulate[1]:LIMit:CLEar:AUTO <boolean> | ONCE

This command controls when the FCO (fail counter) is cleared of any limit failures. The FCO is used to determine the results returned by the CALCulate[1]:LIMit:FAIL? query.

- If ON is specified, the FCO is set to 0 each time a measurement is:
 - Initiated using INITiate[:IMMEDIATE]
 - Initiated using INITiate:CONTinuous ON
 - Measured using MEASure?
 - Read using READ?
- If OFF is specified, the FCO is not cleared by the above commands.
- If ONCE is specified, the FCO is cleared only after the first initialization then starts accumulating any limit failures.

SyntaxExample



CALC1:LIM:CLE:AUTO 1

This command switches on automatic clearing of the FCO for the current window/measurement.

Reset Condition

On reset, measurement is set to ON.

Query

```
CALCulate[1]:LIMit:CLEar:AUTO?
```

The query command enters a 1 or 0 into the output buffer indicating whether limit failures are cleared automatically when a new measurement is initiated on the current window section.

- 1 is entered into the output buffer when limit failures are cleared automatically when a new measurement is initiated.
- 0 is entered into the output buffer when limit failures are not cleared automatically when a new measurement is initiated.

In the case where limit failures are cleared once, when a query occurs a 1 is entered into the output buffer if no measurement is initiated. If a measurement is initiated then 0 is entered.

Query Example

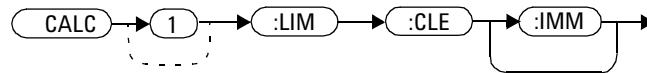
```
CALC1:LIM:CLE:AUTO?
```

This command queries when the FCO is cleared for the current window/measurement.

CALCulate[1]:LIMit:CLEar[:IMMEDIATE]

This command immediately clears the FCO (fail counter) of any limit failures for the current window. The FCO is used to determine the results returned by the CALCulate[1]:LIMit:FAIL? query.

Syntax



Example

```
CALC1:LIM:CLE:IMM
```

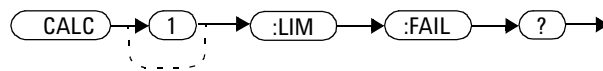
This command clears the FCO for the current window/measurement.

CALCulate[1]:LIMit:FAIL?

This query enters a 1 or 0 into the output buffer indicating whether there have been any limit failures for the current window. A limit failure is defined as `CALC[1]:LIMit:FCO?` being non-zero. The FCO (fail counter) can be zeroed using the `CALC[1]:LIMit:CLEar` command.

- 1 is returned when one or more limit failures have occurred
- 0 is returned when no limit failures have occurred

Syntax



Example

```
CALC1:LIM:FAIL?
```

This command queries if there have been any limit failures on the current window/measurement.

Reset Condition

On reset, the buffer is set to zero for the current window measurements.

CALCulate[1]:LIMit:FCOunt?

This query returns the total number of limit failures for the current window/measurement.

If the appropriate STATE commands are set to ON, each time a measurement is initiated on the current window/measurement and the result is outside the limits, the counter is incremented by one.

If the measured value is equal to a limit, this is a limit pass.

The counter is reset to zero by any of the following commands:

- *RST
- CALCulate[1]:LIMit:CLEar:IMMediate
- CALCulate[1]:LIMit:CLEar:AUTO ON

When CALCulate[1]:LIMit:CLEar:AUTO is set to ON, the counter is set to zero *each* time a measurement is:

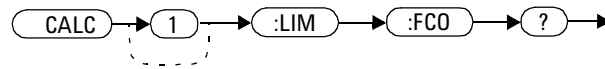
- measured using MEASure?
- read using READ?
- initiated using:
 - INITiate[:IMMediate] or,
 - INITiate:CONTinuous ON

When CALCulate[1]:LIMit:CLEar:AUTO is set to ONCE, the counter is set to zero the *first* time a measurement is:

- measured using MEASure?
- read using READ?
- initiated using:
 - INITiate[:IMMediate] or,
 - INITiate:CONTinuous ON

The maximum number of errors is $2^{16}-1$. If more than $2^{16}-1$ errors are detected the counter returns to zero.

Syntax



Example

```
CALC1:LIM:FCO?
```

This command queries the number of limit failures on the current window/measurement.

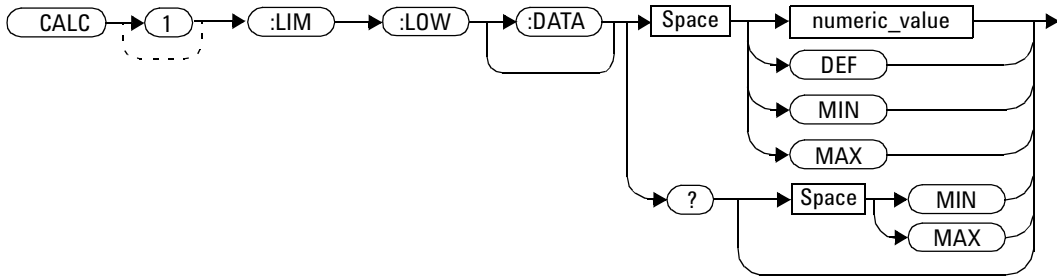
Reset Condition

On reset, the counter is set to zero for the current window/measurement.

CALCulate[1]:LIMit:LOWer[:DATA] <numeric_value>

This command enters a value for the lower test limit for the current window/measurement used in the CALCulate[1]:LIMit:FAIL? test. The units used are dependent on the current setting of UNIT:POWer. When the measured value is less than the value specified in CALCulate[1]:LIMit:LOWer[:DATA], CALCulate[1]:LIMit:FAIL? reports a fail. When the measured value is greater than or equal to the limit, a fail is not reported.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value for the lower test limit: <ul style="list-style-type: none"> DEF: the default is -90.00 dBm or -90 db MIN: -150 dBm or -180 dB MAX: +230 dBm or +200 dB 	-150 to +230 dBm or -180 to +200 dB DEF MIN MAX

Example

```
CALC1:LIM:LOW:DATA 0.1
```

This command enters a lower limit for the current window/measurement depending on the window's units as follows:

dBm = 0.1 dBm

W = 100 mW

dB = 0.1 dB

% = 0.1 %

Reset Condition

On reset, the current window/measurements are set to -90.00 dBm or -90 dB (DEF).

Query

```
CALCulate[1]:LIMit:LOWer[:DATA]? [MIN|MAX]
```

The query returns the current setting of the lower limit or the values associated with MIN and MAX for the current window.

Query Example

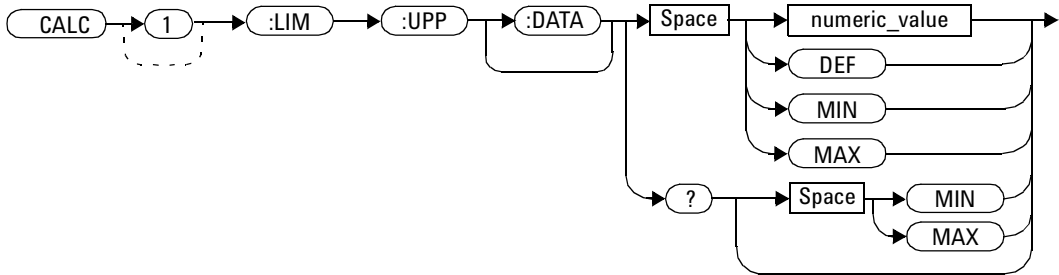
```
CALC1:LIM:LOW:DATA?
```

This command queries the lower limit set for the current window/measurement.

CALCulate[1]:LIMit:UPPer[:DATA] <numeric_value>

This command enters a value for the upper test limit for the current window/measurement used in the CALCulate[1]:LIMit :FAIL? test. The units used are dependent on the current setting of UNIT:POWer. When the measured power is greater than the value specified in CALCulate[1]:LIMit:UPPer[:DATA], CALCulate[1]:LIMit:FAIL? reports a fail. When the measured level is less than or equal to the limit, a fail is not reported.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value for the lower test limit: <ul style="list-style-type: none"> • DEF: the default is -90.00 dBm or -90 db • MIN: -150 dBm or -180 dB • MAX: +230 dBm or +200 dB 	-150 to +230 dBm or -180 to +200 dB DEF MIN MAX

Example

```
CALC1:LIM:UPP:DATA 5
```

This command enters an upper limit for the current window/measurement depending on the window's units as follows:

dBm = 5 dBm
W = 5 W
dB = 5 dB
% = 5 %

Reset Condition

On reset, the measurement is set to +90.00 dBm or +90 dB.

Query

```
CALCulate[1]:LIMit:UPPer[:DATA]? [MIN|MAX]
```

Query Example

CALC1:LIM:UPP:DATA?

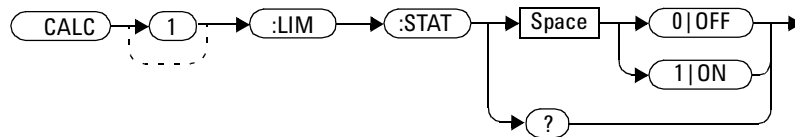
This command queries the setting of the upper limit for the current window/measurement.

The query returns the current setting of the upper limit or the values associated with MIN and MAX for the current window/measurement.

CALCulate[1]:LIMit:STATe <boolean>

This command enables/disables the test limits for the current window/measurement.

Syntax



Example

```
CALC1:LIM:STAT 1
```

This command enables the limit checking function for the current window/measurement.

Reset Condition

On reset, limit checking is disabled.

Query

```
CALCulate[1]:LIMit:STATe?
```

The query enters 1 or 0 into the output buffer indicating the status of the limits testing feature for the current window/measurement.

- 1 is returned when limits testing is enabled
- 0 is returned when limits testing is disabled

Query Example

```
CALC1:LIM:STAT?
```

This command queries whether the limit checking function for the current window/measurement is on or off.

Error Message

If CALCulate[1]:LIMit:STATe is set to ON while [SENSe[1]]:MRATe is set to FAST, error -221, “Settings Conflict” occurs.

CALCulate[1]:MATH Commands

These commands define and carry out the following mathematical transformations on SENSE data:

- Single measurement

The following commands are detailed in this section:

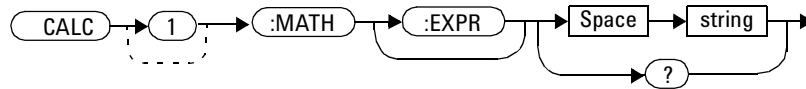
```
CALCulate[1]:MATH[:EXPRession] <string>
```

```
CALCulate[1]:MATH[:EXPRession]:CATalog?
```

CALCulate[1]:MATH[:EXPRession] <string>

This command sets the current window/measurement to a single measurement.

Syntax



Parameters

Item	Description/Default	Range of Values
string	A single string value detailing the measurement type: <ul style="list-style-type: none"> The default is SENS1. 	"(SENS1)" ¹

¹ Quotes are mandatory. Either single or double quotes may be used.

Example

```
CALC1:MATH "(SENS1)"
```

This command sets the current window/measurement to make a single measurement.

Reset Condition

On reset, the current window measurements are set to "(SENS1)".

Query

```
CALCulate[1]:MATH[:EXPRession]?
```

The query returns the current math measurement setting on the current window/measurement.

Query Example

```
CALC1:MATH?
```

This command queries the current setting of the math expression on the current window/measurement.

Error Messages

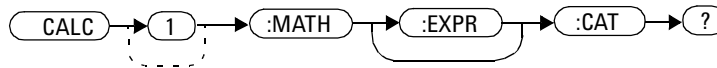
- For single measurement, if <string> is not set to "(SENS1)" while [SENSe[1]):MRATe is set to FAST, error -221, "Settings Conflict" occurs.

CALCulate[1]:MATH[:EXPRession]:CATalog?

This query lists all the defined expressions. The response is a list of comma separated strings. Each string contains an expression.

- For single measurement the string is:
" (SENS1) "

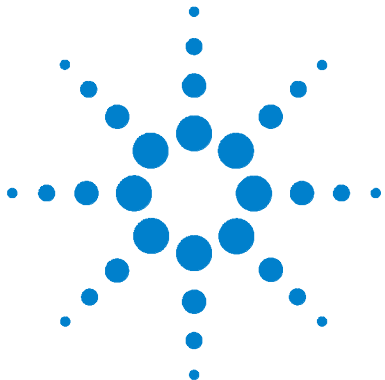
Syntax



Example

CALC1:MATH:CAT?

This command lists all the defined math expressions.



4 CALibration Subsystem

CALibration Subsystem	108
CALibration[1][:ALL]	109
CALibration[1][:ALL]?	110
CALibration[1]:AUTO [ONCE ON OFF 0 1]	112
CALibration:ZERO:AUTO [ONCE ON OFF 0 1]	114
CALibration[1]:ZERO:TYPE <EXTernal INTernal>	117

This chapter explains how the CALibration command subsystem is used to zero and calibrate the power sensor.



CALibration Subsystem

The CALibration command subsystem is used to zero the power sensor.

Zeroing of the power sensor is recommended:

- When a 5 °C change in temperature occurs
- When you change the power sensor
- Every 24 hours
- Prior to measuring low level signals. For example, 10 dB above the lowest specified power for your sensor.

The following CALibration commands are overlapped commands:

- CAL:ALL
- CAL:AUTO
- CAL:ZERO:AUTO

An overlapped command allows the instrument to continue parsing and executing subsequent commands while it is still executing.

Keyword	Parameter Form	Notes	Page
CALibration			
[:ALL]		[event; no query]	page 109
[:ALL]?		[event; query]	page 110
:AUTO	<boolean> ONCE		page 112
:ZERO			
:AUTO	<boolean> ONCE		page 114
:TYPE	<EXTernal INTernal>		page 117

CALibration[1][:ALL]

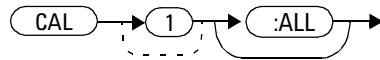
NOTE

This command is identical to `CALibration[1][:ALL]?`, however, unlike the query it does not provide a response to indicate whether the calibration has been successful or not.

This command causes the power sensor to perform a zeroing sequence. The zeroing sequence consists of:

- 1 Zeroing the power sensor (`CALibration:ZERO:AUTO ONCE` or `CALibration:AUTO ONCE`).

Syntax



Example

`CAL1:ALL`

This command causes the power sensor to perform a zeroing sequence.

Error Messages

- If zeroing was not carried out successfully the error -231, “Data Questionable; ZERO ERROR” occurs.

CALibration[1][:ALL]?

NOTE

This query is identical to `CALibration[1] [:ALL]`, however, unlike the command, it provides a response to indicate whether the calibration has been successful or not.

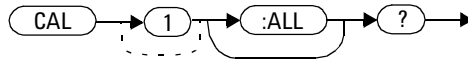
This query causes the power sensor to perform a zeroing sequence. The zeroing sequence consists of:

- 1 Zeroing the power sensor (`CALibration:ZERO:AUTO ONCE` or `CALibration:AUTO ONCE`)

When the zeroing sequence is completed, 0 or 1 is entered into the output buffer to indicate if the sequence was successful. If the result is:

- 0, the zeroing has passed
- 1, the zeroing has failed

Syntax



Query Example

```
CAL1:ALL?
```

This command causes the power sensor to perform a zeroing sequence and return a result.

Error Messages

- If zeroing was not carried out successfully the error -231, “Data Questionable; ZERO ERROR” occurs.

CALibration[1]:AUTO [ONCE | ON | OFF | 0 | 1]

This command zeroes the power sensor when enabled.

When ONCE is enabled, the zeroing will perform only once for both internal and external zeroing type.

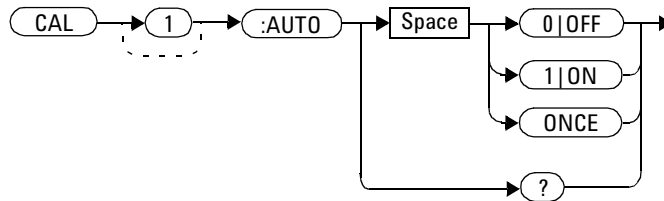
When 1|ON is enabled, the zero is updated if the sensor's temperature changes by ± 5 °C or the time since last zeroing is greater than 24 hours during internal zeroing. The zero is maintained by *on-the-fly* zero measurements.

The 0|OFF parameter is only required for the query response and is ignored in the command.

NOTE

This command performs the same function as CALibration:ZERO:AUTO [ONCE | ON | OFF | 0 | 1].

Syntax



Example

```
CAL1:AUTO ONCE
```

This command causes the power sensor to perform a zeroing.

Reset Condition

On reset, automatic calibration is enabled.

Query

```
CALibration[1]:AUTO?
```

The query always returns a value of 0 when the event is successful.

Error Messages

- If the zeroing was not carried out successfully the error -231, “Data Questionable; CAL ERROR” occurs.

CALibration:ZERO:AUTO [ONCE | ON | OFF | 0 | 1]

This command zeroes the power sensor when enabled.

When ONCE is enabled, the zeroing will perform only once for both internal and external zeroing type.

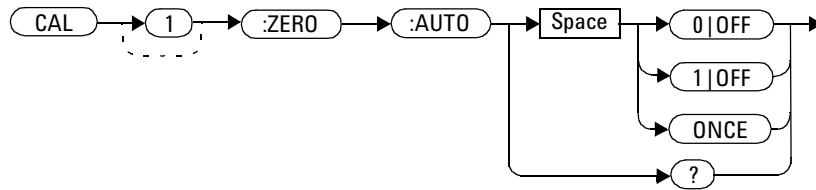
When 1|ON is enabled, the zero is updated if the sensor's temperature changes by ± 5 °C or the time since last zeroing is greater than 24 hours during internal zeroing. The zero is maintained by *on-the-fly* zero measurements.

The 0|OFF parameter is only required for the query response and is ignored in the command.

NOTE

This command performs the same function as CALibration[1]:AUTO [ONCE | ON | OFF | 0 | 1].

Syntax



Example

```
CAL1 :ZERO:AUTO ONCE
```

This command causes the power sensor to perform a zeroing routine.

Reset Condition

On reset, automatic zeroing is enabled.

Query

```
CALibration[1]:ZERO:AUTO?
```

The query always returns a value of 0 when the event is successful.

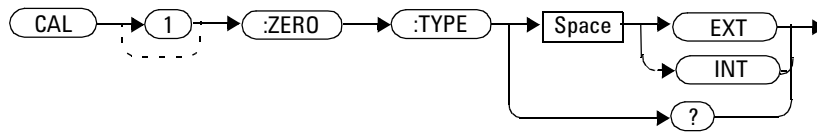
Error Messages

- If zeroing was not carried out successfully the error -231, “Data Questionable; ZERO ERROR” occurs.

CALibration[1]:ZERO:TYPE <EXTernal | INTernal>

This command is used to configure the power sensor either for enternal zeroing or internal zeroing.

Syntax



Example

```
CAL:ZERO:TYPE EXT
```

This command changes the type of zeroing to external.

Reset Condition

On reset, the zeroing type will not be affected.

Query

```
CALibration[1]:ZERO:TYPE?
```

This query returns the type of current zeroing either EXT or INT. This format is in string.

Query Example

```
CAL:ZERO:TYPE?
```

This command queries the type of zeroing for the sensor .

Error Messages

- This command is able to configure zeroing type to “EXT” and “INT” only. The error -231, “Invalid character type” occurs for any other values.



5 FORMat Subsystem

FORMat Subsystem [120](#)

FORMat[:READings]:BORDER <character_data> [121](#)

FORMat[:READings][:DATA] <character_data> [123](#)

This chapter explains how the FORMat subsystem is used to set a data format for transferring numeric information.



FORMat Subsystem

The FORMat subsystem sets a data format for transferring numeric information. This data format is used only for response data by commands that are affected by the FORMat subsystem.

The queries affected are:

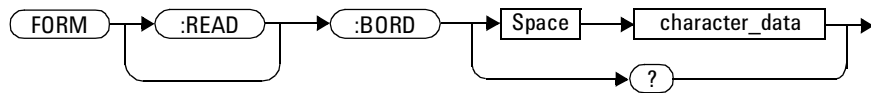
- FETCh?
- READ?
- MEASure?

Keyword	Parameter Form	Notes	Page
FORMat			
[:READings]			
:BORDer	<character_data>		page 121
[:DATA]	<character_data>		page 123

FORMat[:READings]:BORDER <character_data>

This command controls whether the binary data is transferred in normal or swapped Byte ORDER. It is only used when FORMat[:READings][:DATA] is set to REAL.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Byte order of binary data transfer: <ul style="list-style-type: none"> • NORMAl • SWAPped 	NORMAl SWAPped

Example

FORM:BORD SWAP

This command sets the byte order to swapped.

Reset Condition

On reset, this value is set to NORMAl.

Query

```
FORMat[:READings]:BORDER?
```

The query returns the current setting of the byte order. The format of the response is NORMAL or SWAPPED..

Query Example

```
FORM:BORD?
```

This command queries the current byte order setting.

FORMat[:READings][:DATA] <character_data>

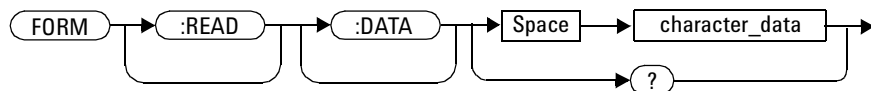
This command sets the data format for transferring numeric information to either ASCII or REAL:

- When the format type is ASCII, numeric data is output as ASCII bytes in the <NR3> format.
- When the format type is REAL, numeric data is output as IEEE 754 64 bit floating point numbers in a definite length block. The result is an 8 byte block per number. Each complete block is terminated by a line feed character.

NOTE

FORMat data formatting is not affected by TRACe subsystem data formatting.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Data format for transferring data: <ul style="list-style-type: none"> • ASCII • REAL 	ASCII REAL

Example

```
FORM REAL
```

This command sets the format to REAL.

Reset Condition

On reset, the format is set to ASCii.

Query

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

The query returns the current setting of format: either ASCii or REAL.

Query Example

```
FORM?
```

This command queries the current format setting.



6 MEMory Subsystem

MEMory Subsystem	126
MEMory:CATalog Commands	128
MEMory:CATalog[:ALL]?	129
MEMory:CATalog:STATe?	131
MEMory:CATalog:TABLE?	132
MEMory:CLEar Commands	134
MEMory:CLEar[:NAME] <character_data>	135
MEMory:CLEar:TABLE	137
MEMory:FREE Commands	138
MEMory:FREE[:ALL]?	139
MEMory:FREE:STATe?	140
MEMory:FREE:TABLE?	141
MEMory:NSTates?	142
MEMory:STATe Commands	143
MEMory:STATe:CATalog?	144
MEMory:STATe:DEFine <character_data>,<numeric_value>	145
MEMory:TABLE Commands	147
MEMory:TABLE:FREQUency <numeric_value>{,<numeric_value>}	148
MEMory:TABLE:FREQUency:POINts?	151
MEMory:TABLE:GAIN[:MAGNitude] <numeric_value>{,<numeric_value>}	152
MEMory:TABLE:GAIN[:MAGNitude]:POINts?	155
MEMory:TABLE:MOVE <character_data>,<character_data>	156
MEMory:TABLE:SElect <character_data>	158

This chapter explains how the MEMory command subsystem is used to create, edit and review sensor calibration tables.



MEMory Subsystem

The MEMory command subsystem is used to:

- Edit and review sensor frequency dependent offset tables.
- Store sensor frequency dependent offset tables.
- Edit and review sensor save/recall registers.

Stored tables remain in the power sensor's memory during power down. The power sensor is capable of storing 10 frequency dependent offset tables of 80 frequency points each.

Keyword	Parameter Form	Notes	Page
MEMory			
:CATalog			
[:ALL]?		[query only]	page 129
:STATe?		[query only]	page 131
:TABLE?		[query only]	page 132
:CLEar			
[:NAME]	<character_data>	[no query], [non-SCPI]	page 135
:TABLE		[no query]	page 137
:FREE			
[:ALL]?		[query only]	page 139
:STATe?		[query only]	page 140
:TABLE?		[query only]	page 141
:NSTates?		[query only]	page 142
:STATe			
:CATalog?		[query only]	page 144
:DEFine	<character_data> [, <numeric_value>]	[non-SCPI]	page 145
:TABLE			
:FREQuency	<numeric_value> [, <numeric_value>]		page 148
:POINTs?		[query only]	page 151

Keyword	Parameter Form	Notes	Page
:GAIN			
	[:MAGNitude]	<numeric_value> [, <numeric_value>]	[non-SCPI] page 152
	:POINTs?	[query only], [non-SCPI]	page 155
:MOVE	<character_data>, <character_data>	[no query], [non-SCPI]	page 156
:SElect	<character_data>	[no query], [non-SCPI]	page 158

MEMory:CATalog Commands

These commands are used to query information on the current contents of a power sensor's:

- Frequency dependent offset tables
- Save/recall registers

The following commands are detailed in this section:

MEMory:CATalog[:ALL]?

MEMory:CATalog:STATE?

MEMory:CATalog:TABLE?

MEMory:CATalog[:ALL]?

This command lists stored frequency dependent offset tables and save/recall registers.

The power sensor returns the data in the form of two numeric parameters and as many strings as there are stored tables and save/recall registers:

```
<numeric_value>, <numeric_value>{, <string>}
```

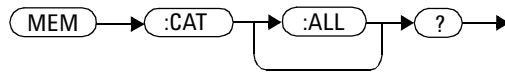
- The first numeric parameter indicates the amount of memory, in bytes, used for the storage of tables and registers.
- The second numeric parameter indicates the memory, in bytes, available for the storage of tables and registers.
- Each string parameter returned indicates the name, type and size of a stored table or save/recall register:
 - <string>, <type>, <size>
 - <string> indicates the name of the table or save/recall register.
 - <type> indicates TABL for frequency dependent offset tables, or STAT for a save/recall register.
 - <size> indicates the size of the table or save/recall register in bytes.

A sample of a response may look like the following:

```
1178,26230,"DEFAULT,TABL,14","TABLE1,TABL,116",
"TABLE2,TABL,74",....."State0,STAT,1619",
"State1,STAT,1619","State2,STAT,1619" .....
```

There are also ten frequency dependent offset tables named CUSTOM_A through CUSTOM_J which do not contain any data when the power sensor is shipped from the factory.

Syntax



Example

MEM:CAT?

This command queries the list of tables and save/recall registers.

MEMory:CATalog:STATe?

This command is used to list the save/recall registers.

The power sensor returns the data in the form of two numeric parameters and as many strings as there are save/recall registers.

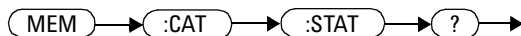
<numeric_value>,<numeric_value>{,<string>}

- The first numeric parameter indicates the amount of memory, in bytes, used for the storage of registers.
- The second parameter indicates the memory, in bytes, available for the storage of registers.
- Each string parameter returned indicates the name, type and size of a save/recall register:
 - <string>,<type>,<size>
 - <string> indicates the name of the save/recall register.
 - <type> indicates *STAT* for save/recall register.
 - <size> indicates the size of the save/recall register in bytes.

For example, a sample of a response may look like:

0,16190,"State0,STAT,0","State1,STAT,0"

Syntax



Example

MEM:CAT:STAT?

This command queries the list of save/recall registers.

MEMory:CATalog:TABLE?

This command is used to list the stored frequency dependent offset tables.

The power sensor returns the data in the form of two numeric parameters and as many strings as there are stored tables.

```
<numeric_value>,<numeric_value>{,<string>}
```

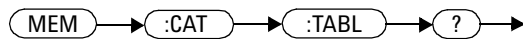
- The first numeric parameter indicates the amount of memory, in bytes, used for the storage of tables.
- The second parameter indicates the memory, in bytes, available for the storage of tables.
- Each string parameter returned indicates the name, type and size of a stored table:
 - <string>,<type>,<size>
 - <string> indicates the name of the table.
 - <type> indicates TABL for a table.
 - <size> indicates the size of the table in bytes.

For example, a sample of a response may look like:

```
1178,10040,"DEFAULT,TABL,14","TABLE1,TABL,116",
"TABLE2,TABL,74","TABLE3,TABL,62".....
```

There are also ten frequency dependent offset tables named CUSTOM_A through CUSTOM_J which do not contain any data when the power sensor is shipped from the factory.

Syntax



Example

MEM:CAT:TABL?

This command queries the list of stored tables.

MEMory:CLEar Commands

These commands are used to remove the frequency dependent offset tables and save/recall registers. This subsystem removes the data contents but does not affect the name of the associated table or save/recall register.

The following commands are detailed in this section:

```
MEMory:CLEar:[NAME] <character_data>
```

```
MEMory:CLEar:TABLE
```

NOTE

The contents cleared using these commands are non-recoverable.

MEMory:CLEar[:NAME] <character_data>

This command clears the contents of a specified frequency dependent offset table, or save/recall register.

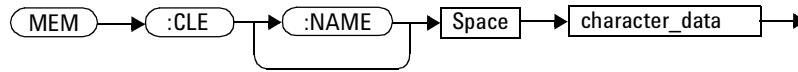
Although the table remains, a `MEMory:TABLE:FREQuency|GAIN:POINTs?` query returns a 0 as there are no contents in the table.

For frequency dependent offset tables, this command is an alternative form of the `MEMory:CLEar:TABLE` command, the only difference being the method in which the table is selected.

NOTE

The contents cleared using this command are non-recoverable.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Contains an existing table name or save/recall register.	Any existing table name or save/recall register.

Example

```
MEM:CLEAR "TABLE5"
```

This command clears the contents of frequency dependent offset table, TABLE5.

Error Messages

If the table or save/recall register name does not exist, error -224, “Illegal parameter value” occurs.

MEMory:CLEar:TABLE

This command is used to clear the contents of the table currently selected using `MEMory:TABLE:SElect`. Although the table remains, a `MEMory:TABLE:FREQuency|GAIN:POINts?` query returns a 0 as the table contents are empty.

This command is an alternative form of the `MEMory:CLEar[:NAME]` command. The difference is the method in which the table is selected.

NOTE

The contents cleared using this command are non-recoverable.

Syntax



Example

```
MEM:CLE:TABLE
```

This command clears the contents of the currently selected table.

Error Message

If no table is selected, error -221, “Settings conflict” occurs.

MEMory:FREE Commands

These commands are used to return information on the amount of free memory space available for frequency dependent offset tables and save/recall registers.

The following commands are described in this section:

MEMory:FREE[:ALL]?

MEMory:FREE:STATE?

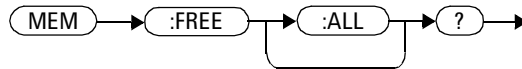
MEMory:FREE:TABLE?

MEMory:FREE[:ALL]?

This query returns the amount of memory free for frequency dependent offset tables, and save/recall registers. The format of the response is:

<bytes_available>, <bytes_in_use>

Syntax



Example

MEM:FREE?

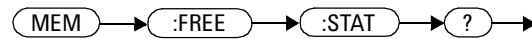
This command queries the amount of free memory in total.

MEMory:FREE:STATe?

This query returns the amount of memory free for save/recall registers.
The format of the response is:

<bytes_available>, <bytes_in_use>

Syntax



Example

MEM:FREE:STAT?

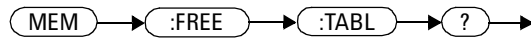
This command queries the amount of free memory for save/recall registers.

MEMory:FREE:TABLE?

This query returns the amount of memory free for frequency dependent offset tables. The format of the response is:

<bytes_available>, <bytes_in_use>

Syntax



Example

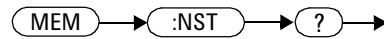
MEM:FREE:TABLE?

This command queries the amount of free memory for tables.

MEMory:NStates?

This query returns the number of registers that are available for save/recall. As there are ten registers this query always returns ten.

Syntax



Example

MEM:NST?

This command queries the number of registers available for save/recall.

MEMory:STAtE Commands

These commands are used to query and define register names.

The following commands are described in this section:

MEMory:STAtE:CAtalog?

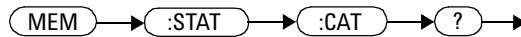
MEMory:STAtE:DEFine

MEMory:STATe:CATalog?

This query returns a list of the save/recall register names in ascending order of register number. The format of the response is:

<string>,<string>,...,<string>

Syntax



Example

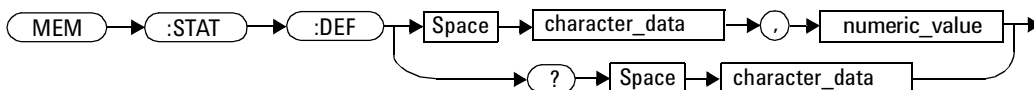
MEM:STAT:CAT?

This command queries the register names.

MEMory:STATe:DEFine <character_data>,<numeric_value>

This command is used to associate a name with a save/recall register number.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the register name. A maximum of 12 characters can be used.	A to Z (uppercase) a to z (lowercase) 0-9 _ (underscore)
numeric_value	A numeric value (<NRf>) for the register number.	0 to 9

Example

MEM:STAT:DEF "SETUP1",4 *This command names register 4 SETUP1.*

Query

MEMory:STATe:DEFine? <string>

The query returns the register number for the given register name.

Query Example

MEM:STAT:DEF? "SETUP1"

This command queries the register number of SETUP1.

Error Messages

- If the register number is out of range, error -222, “Data out of range” occurs.
- If the name is invalid, error -224, “Illegal parameter value” occurs.
- If a register with the same name already exists, error -257, “File name error” occurs (command only).

MEMory:TABLE Commands

These commands are used to define a frequency dependent offset table, and to write to and read data from it.

The following commands are described in this section:

```
MEMory:TABLE:FREQuency <numeric_value>{,<numeric_value>}
```

```
MEMory:TABLE:FREQuency:POINTs?
```

```
MEMory:TABLE:GAIN[:MAGNitude]  
<numeric_value>{,<numeric_value>}
```

```
MEMory:TABLE:GAIN[:MAGNitude]:POINTs?
```

```
MEMory:TABLE:MOVE <character_data>,<character_data>
```

```
MEMory:TABLE:SElect <character_data>
```

MEMory:TABLE:FREQuency <numeric_value>{,<numeric_value>}

This command is used to enter frequency data into the current selected table. Any previous frequency list is cleared before the new frequency list is stored. The frequencies must be entered in ascending order. Entries in the frequency lists correspond as shown in [Table 6-1](#) with entries in the offset factor lists.

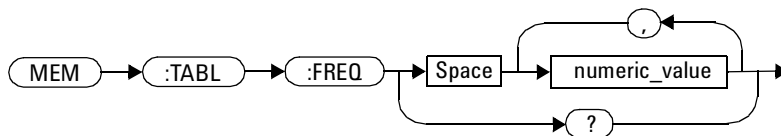
Table 6-1 Frequency and Offset Factor List

Frequency	Offset
Frequency 1	Offset 1
"	"
Frequency 80	Offset 80

Ensure that the frequency points you use cover the frequency range of the signals that you want to measure. If you measure a signal with a frequency outside the frequency range defined in the table, then the power sensor uses the highest or lowest point in the table to calculate the calibration factor/offset.

Depending on available memory, the power sensor is capable of storing 10 frequency dependent offset tables, each containing 80 points.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value for the frequency. The default units are Hz.	1 kHz to 1000.0 GHz ^{1,2}

¹ The following measurement units can be used:

Hz

kHz (10^3)

MHz (10^6)

GHz (10^9)

² All frequencies are truncated to a multiple of 1 kHz.

Example

```
MEM:TABLE:FREQ  
200kHz,600kHz
```

This command enters frequencies of 200 kHz and 600 kHz into the currently selected table.

Query

```
MEMory:TABLE:FREQuency?
```

The query returns a list of frequency points for the table currently selected. The frequencies are returned in Hz.

Query Example

```
MEM:TABLE:FREQ?
```

This command queries the frequency points in the currently selected table.

Error Messages

- If more than 80 frequencies are in the list, error -108, “Parameter not allowed” occurs.
- If the frequencies are not entered in ascending order, error -220, “Parameter error;Frequency list must be in ascending order” occurs.
- If a table has not been specified using the MEMory:TABLE:SElect command, the data cannot be entered into the table and error -221, “Settings conflict” occurs.
- If a frequency is set which is outside of the allowed frequency range, error -222, “Data out of range” occurs.

MEMory:TABLE:FREQuency:POINts?

This query returns the number of frequency points for the table currently selected. The response format is <NRf>. If no frequency values have been set, this command returns 0. If no table is selected, this command returns NAN.

Syntax



Example

MEM:TABLE:FREQ:POIN?

This command queries the number of frequency points in the current table.

MEMory:TABLE:GAIN[:MAGNitude] <numeric_value>{,<numeric_value>}

This command is used to enter offsets into the frequency dependent offset table, currently selected using `MEMory:TABLE:SELEct`. Any previous offset list is cleared before the new offsets are stored.

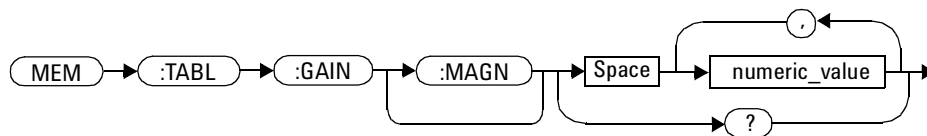
A maximum of 80 parameters for frequency dependent offset tables can be sent with this command.

Entries in the frequency lists correspond as shown in [Table 6-2](#) with entries in the offset factor lists.

Table 6-2 Frequency and Offset Factor List

Frequency	Offset
Frequency 1	Offset 1
"	"
Frequency 80	Offset 80

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value for the calibration/ offset factors. The units are PCT.	1.0 to 150.0

Example

```
MEM:TABLE:SEL "Sensor_1"      This command enters a reference offset
MEM:TABLE:GAIN 97,99.5,97.4 factor of 97 %, 99.5 % and 97.4 % into the
                               sensor frequency dependent offset table.
```

Query

```
MEMory:TABLE:GAIN[:MAGNitude]?
```

The query returns a list of offset points for the currently selected table.

Query Example

```
MEM:TABLE:GAIN?
```

This command queries the offset in the current table.

Error Messages

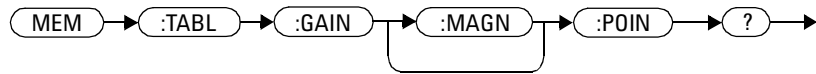
- If more than 80 offsets for frequency dependent offset tables are in the list, error -108, “Parameter not allowed” occurs.
- If a table is not specified using the MEMory:TABLE:SElect command, the data cannot be entered and error -221, “Settings conflict” occurs.
- If any of the offset factors are outside of the allowed range, error -222, “Data out of range” occurs.

MEMory:TABLE:GAIN[:MAGNitude]:POINts?

This query is used to return the number of offset points for the currently selected table.

If no values have been set, 0 is returned. If no table is selected, NAN is returned.

Syntax



Example

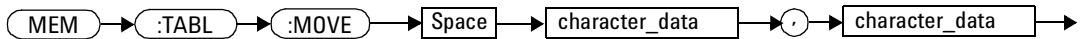
MEM:TABLE:GAIN:POIN?

This command queries the number of offset points in the current table.

MEMory:TABLE:MOVE <character_data>,<character_data>

This command is used to rename a frequency dependent offset table.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data (1st parameter)	Contains the existing table name.	existing table name
character_data(2nd parameter)	Details the new table name. A maximum of 12 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9 _ (underscore)

Example

```
MEM:TABLE:MOVE
"tab1", "tab1a"
```

This command renames a table named tab1 to tab1a.

Error Messages

- If either table name is invalid, error -224, “Illegal parameter value”

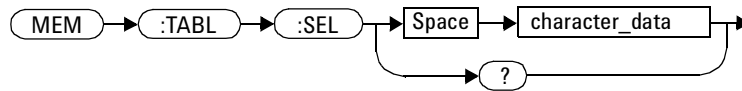
occurs.

- If the first parameter does not match an existing table name, error -256, “File name not found” occurs.
- If the second parameter matches an existing table name or save/recall register, error -257, “File name error” occurs.

MEMory:TABLE:SElect <character_data>

This command is used to activate either a frequency dependent offset table. A table must be activated before any operation can be performed on it.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the new table name. A maximum of 12 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9 _ (underscore)

Example

```
MEM:TABLE:SEL "Sensor1"
```

This command selects a frequency dependent offset table named "Sensor1".

Query

```
MEMory:TABLE:SElect?
```

The query returns the name of the currently selected table.



7 SENSe Subsystem

[SENSe] Subsystem	160
SENSe[1]:AVERage Commands	162
SENSe[1]:AVERage:COUNT <numeric_value>	163
SENSe[1]:AVERage:COUNT:AUTO <boolean>	166
SENSe[1]:AVERage:SDEtect <boolean>	169
SENSe[1]:AVERage[:STATe] <boolean>	171
SENSe[1]:CORRection:CSET[2]Commands	173
SENSe[1]:CORRection:CSET[2]Commands	173
SENSe[1]:CORRection:CSET[2][:SElect] <string>	174
SENSe[1]:CORRection:CSET[2]:STATe <boolean>	176
[SENSe[1]]:CORRection:FDOffset GAIN4[:INPut][:MAGNitude]?	178
[SENSe[1]]:CORRection:GAIN2 Commands	179
[SENSe[1]]:CORRection:GAIN2:STATe <boolean>	180
[SENSe[1]] SENSe2:CORRection:GAIN2[:INPut][:MAGNitude] <numeric_value>	182
SENSe[1]:DETEctor:FUNCTion <character_data>	185
SENSe[1]:FREQuency[:CW :FIXed] <numeric_value>	187
SENSe[1]:MRATe <character_data>	190
SENSe[1]:POWer:AC:RANGe <numeric_value>	193
SENSe[1]:POWer:AC:RANGe:AUTO <boolean>	195
SENSe[1]:TEMPerature?	197

This chapter explains how the `SENSe` command subsystem directly affects device specific settings used to make measurements.



[SENSE] Subsystem

The SENSE command subsystem directly affects device specific settings used to make measurements. The SENSE subsystem is optional since this is the primary function of the power sensor. The high level command CONFIGure uses the SENSE commands to prepare the power sensor for making measurements. At a lower level SENSE enables you to change the following parameters: RANGE, FREQuency, LOSS, CFACator|GAIN1 (calibration factor), GAIN2 (channel offset), and AVERage, without completely re-configuring the power sensor.

The SENSE command subsystem also allows you to select the measurement, a sensor calibration table, and a frequency dependent offset table.

Keyword	Parameter Form	Notes	Page
[SENSe[1]]			
:AVERage			
:COUNT	<numeric_value>		page 163
:AUTO	<boolean>		page 166
:SDETECT	<boolean>	[non-SCPI]	page 169
[:STATe]	<boolean>		page 171
:CORREction			
:CSET[2]			
:SELEct	<string>		page 174
:STATe	<boolean>		page 176
:FSDOFFset GAIN4			
:[INPut]			
[:MAGNitude]?	<numeric_value>		page 178
:GAIN2			
:STATe	<boolean>		page 180
[INPut]			
[:MAGNitude]	<numeric_value>		page 182
:DETEctor			
:FUNCTion	<character_data>		page 185

7 SENSE Subsystem

Keyword	Parameter Form	Notes	Page
:FREQuency			
[:CW :FIXed]	<numeric_value>		page 187
:MRATe	<character_data>		page 190
:POWeR			
:AC			
:RANGe	<numeric_value>	[non-SCPI]	page 193
:AUTO	<boolean>		page 195
:TEMPeRature?		[query only]	page 197

SENSe[1]:AVERAge Commands

These commands control the measurement averaging which is used to improve measurement accuracy. They combine successive measurements to produce a new composite result.

The following commands are detailed in this section:

```
SENSe[1]:AVERAge:COUNT <numeric_value>  
SENSe[1]:AVERAge:COUNT:AUTO <boolean>  
SENSe[1]:AVERAge:SDETECT <boolean>  
SENSe[1]:AVERAge[:STATe] <boolean>
```

SENSe[1]:AVERage:COUNT <numeric_value>

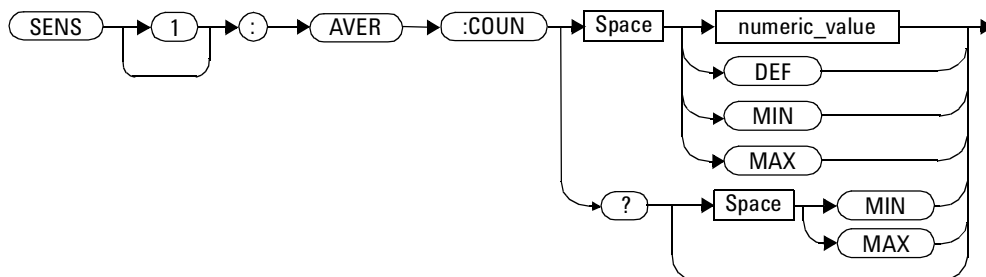
This command is used to enter a value for the filter length. If `SENSe[1]:AVERage:COUNT:AUTO` is set to `ON` then entering a value for the filter length automatically sets it to `OFF`. Increasing the value of filter length increases measurement accuracy but also increases the time taken to make a power measurement.

Entering a value using this command automatically turns the `SENSe[1]:AVERage:STATE` command to `ON`.

NOTE

For most applications, automatic filter length selection (`SENSe[1]:AVERage:COUNT:AUTO ON`) is the best mode of operation. However, manual filter length selection (`SENSe[1]:AVERage:COUNT <numeric_value>`) is useful in applications requiring either high resolution or fast settling times, where signal variations rather than measurement noise need filtering, or when approximate results are needed quickly.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value defining the filter length. DEF: the default value is 4 MIN: 1 MAX: 1024	1 to 1024 DEF MIN MAX

Example

```
AVER:COUN 400
```

This command enters a filter length of 400 for power sensor.

Reset Condition

On reset, the filter length is set to four.

Query

```
AVERage:COUNT? [MIN|MAX]
```

The query returns the current setting of the filter length or the values associated with MIN and MAX. The format of the response is <NR1>.

Query Example

AVER:COUNT?

This command queries the filter length for power sensor.

Error Messages

If a filter length value is entered using SENSE[1]:AVERAGE:COUNT while SENSE[1]:MRATE is set to FAST, the error -221, “Settings Conflict” occurs. However, the filter length value is set but the SENSE[1]:AVERAGE:STATE command is not automatically set ON.

SENSe[1]:AVERage:COUNT:AUTO <boolean>

This command enables and disables automatic averaging. ONCE has no affect on the power sensor.

When the auto filter mode is enabled, the power sensor automatically sets the number of readings averaged together to satisfy the averaging requirements for most power measurements. The number of readings averaged together depends on the resolution and the power level in which the power sensor is currently operating. Figure 7-1 is an example of the averaged number of readings for each range and resolution when the power sensor is in auto measurement average mode.

Setting this command to ON automatically sets the SENSe[1]:AVERage:STATE command to ON.

	Maximum Sensor Power	Resolution Setting			
		1	2	3	4
		1	1	1	8
10 dB		1	1	1	16
10 dB		1	1	2	32
10 dB		1	1	16	256
10 dB		1	8	128	128
	Minimum Sensor Power				

Figure 7-1 is a diagram showing the relationship between Power Sensor Dynamic Range, Resolution Setting, and Number of Averages. The diagram consists of a table with four columns for Resolution Settings (1, 2, 3, 4) and four rows for 10 dB ranges. The top row is labeled 'Maximum Sensor Power' and the bottom row is labeled 'Minimum Sensor Power'. The left side of the table is labeled 'Power Sensor Dynamic Range' and the right side is labeled 'Number of Averages'. The number of averages increases as the resolution setting increases and as the power level decreases (moving from the top row to the bottom row).

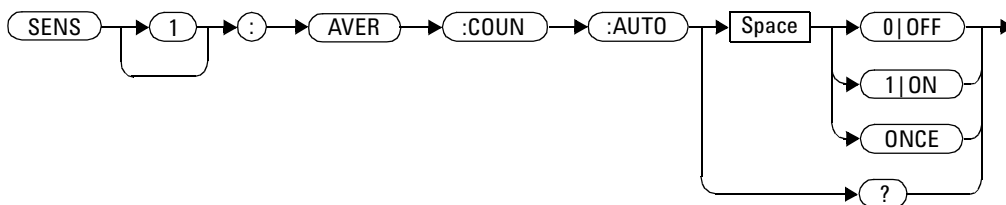
Figure 7-1 Example of Averaged Readings

If SENSe[1]:AVERage:COUNT:AUTO is set to OFF, the filter length is set by the SENSe[1]:AVERage:COUNT command. Using the SENSe[1]:AVERage:COUNT command disables automatic averaging.

Auto averaging is enabled by the MEASure:POWer:AC? and CONFigure:POWer:AC? commands.

NOTE

For most applications, automatic filter length selection (`SENSe[1]:AVERage:COUNT:AUTO ON`) is the best mode of operation. However, manual filter length selection (`SENSe[1]:AVERage:COUNT <numeric_value>`) is useful in applications requiring either high resolution or fast settling times, where signal variations rather than measurement noise need filtering, or when approximate results are needed quickly.

Syntax**Example**

```
AVER:COUN:AUTO OFF
```

This command disables automatic filter length selection for power sensor.

Reset Condition

On reset, automatic averaging is enabled.

Query

SENSe[1]:AVERage:COUNT:AUTO?

The query enters a 1 or 0 into the output buffer indicating whether automatic filter length is enabled or disabled.

- 1 is returned when automatic filter length is enabled
- 0 is returned when automatic filter length is disabled

Query Example

AVER:COUN:AUTO?

This command queries whether automatic filter length selection is on or off for power sensor.

Error Messages

If SENSe[1]:AVERage:COUNT:AUTO is set to ON while SENSe[1]:MRATE is set to FAST, the error -221, “Settings Conflict” occurs. However, automatic averaging is enabled but the SENSe[1]:AVERage:STATE command is not automatically set ON.

SENSe[1]:AVERage:SDETECT <boolean>

This command enables and disables step detection. In AUTO filter mode, the average of the last four values entered into the filter is compared to the average of the entire filter. If the difference between the two averages is greater than 12.5%, the digital filter is cleared. The filter then starts storing new measurement values. This feature shortens the filter time when the input power changes substantially. For the filter output to get to its final value. Note that this result appears to settle faster, although true settling to the final value is unaffected.

NOTE

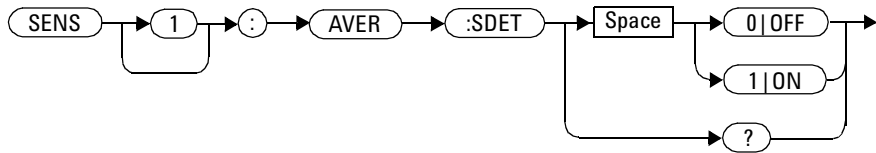
Step detection is automatically disabled when TRIG:DEL:AUTO is ON and the trigger mode is set to free run.

Under these circumstances the value of SENS:AVER:SDET is ignored. Note also that SENS:AVER:SDET is not set by the instrument (that is, SENS:AVER:SDET retains its current setting which may indicate that step detection is ON).

NOTE

With certain pulsing signals step detect may operate on the pulses, preventing the final average from being completed and making the results unstable. Under these conditions SDET should be set to OFF.

Syntax



Example

```
SENS:AVER:SDET OFF
```

This command disables step detection.

Reset Condition

On reset, step detection is enabled.

Query

```
SENSE[1]:AVERage:SDETECT?
```

The query enters a 1 or 0 into the output buffer indicating the status of step detection.

- 1 is returned when step detection is enabled
- 0 is returned when step detection is disabled

Query Example

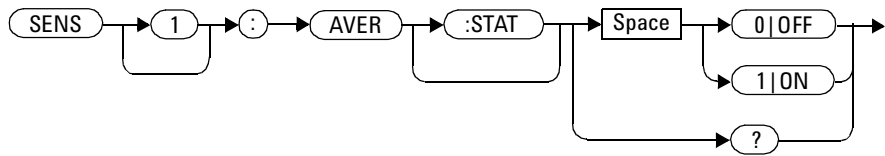
```
SENS:AVER:SDET?
```

This command queries whether step detection is on or off.

SENSe[1]:AVERage[:STATe] <boolean>

This command is used to enable and disable averaging.

Syntax



Example

```
AVER 1
```

This command enables averaging on power sensor.

Reset Condition

On reset, averaging is OFF.

Query

```
SENSe[1]:AVERage[:STATe]?
```

The query enters a 1 or 0 into the output buffer indicating the status of averaging.

- 1 is returned when averaging is enabled
- 0 is returned when averaging is disabled

Query Example

```
SENS: AVER?
```

This command queries whether averaging is on or off for power sensor.

Error Messages

If SENSE[1]:AVERAGE:STATE is set to ON while SENSE[1]: is set to 200, the error -221, “Settings Conflict” occurs.

SENSE[1]:CORREction:CSET[2]Commands

These commands are used to select the active sensor frequency dependent offset table.

The following commands are detailed in this section:

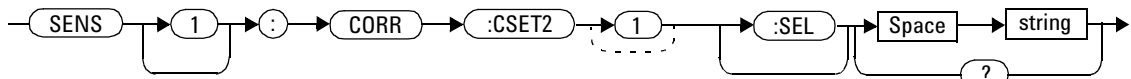
```
SENSE[1]:CORREction:CSET[2][:SElect] <string>  
SENSE[1]:CORREction:CSET[2]:STATE <boolean>
```

SENSe[1]:CORRection:CSET[2][:SElect] <string>

This command enters the name of the sensor frequency dependent offset table which is to be used.

NOTE

If `SENSe[1]:CORRection:CSET2:STATe` is set to `OFF`, the selected sensor frequency offset table is not being used.

Syntax**Parameters**

Item	Description/Default	Range of Values
string	String data representing a sensor frequency dependent offset table name.	Any existing table name (Existing table names can be listed using <code>MEMory:CATalog:TABLE?</code>).

Example

```
CORR:CSET2 `PW1`
```

This command enters the name of the sensor frequency dependent offset table which is to be used.

Reset Condition

On reset the selected table is not affected.

Query

```
SENSE[1]:CORREction:CSET2:[SElect]?
```

The name of the selected table is returned as a quoted string. If no table is selected an empty string is returned.

Query Example

```
CORR:CSET2?
```

This command queries the sensor frequency dependent offset table currently used.

Error Messages

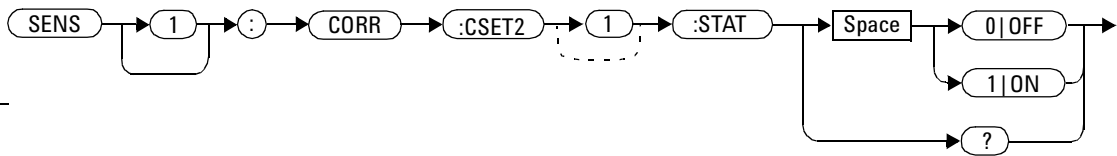
- If <string> is not valid, error -224, “Illegal parameter value” occurs.
- If a table called <string> does not exist, error -256, “File name not found” occurs.
- When a frequency dependent offset table is selected, the power meter verifies that the number of offset points defined is equal to the number of frequency points defined. If this is not the case, error -226, “Lists not the same length” occurs.

SENSe[1]:CORRection:CSET[2]:STATe <boolean>

This command is to enable and disable the use of the currently active sensor frequency dependent offset table (CSET2). When a table has been selected and enabled, the sensor frequency dependent offsets stored in it can be used by specifying the required frequency using the SENSe[1]:FREQuency command.

When the CSET2 command is set to ON, the frequency dependent offset is taken from the sensor frequency dependent offset table and is used.

Syntax



Example

```
CORR:CSET2:STAT 1
```

This command enables the use of the currently active sensor frequency dependent offset table for power sensor.

Reset Condition

On reset, the sensor frequency dependent offset table are not affected.

Query

```
SENSe[1]:CORRection:CSET[2]:STATe?
```

The query returns a 1 or 0 into the output buffer indicating whether a table is enabled or disabled.

- 1 is returned when the table is enabled
- 0 is returned when the table is disabled

Query Example

```
SENS1:CORR:CSET2:STAT?
```

This command queries whether there is currently an active sensor frequency dependent offset table for the sensor.

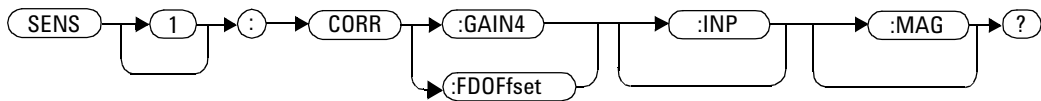
Error Messages

If you attempt to set this command to ON and no table has been selected using `SENSe[1]:CORRection:CSET[2]:[SElect]` then error -221, “Settings conflict” occurs and `SENSe[1]:CORRection:CSET[2]:STATE` remains OFF.

[SENSe[1]]:CORRection:FDOFfset | GAIN4[:INPut][:MAGNitude] ?

This command is used to return the frequency dependent offset currently being applied.

Syntax



Example

CORR:GAIN4?

This command queries the current frequency dependent offset being applied to current measurement.

Reset Condition

On reset, the frequency dependent offset is not affected.

[SENSE[1]]:CORRection:GAIN2 Commands

These commands provide a simple correction to a measurement for an external gain/loss.

The following commands are detailed in this section:

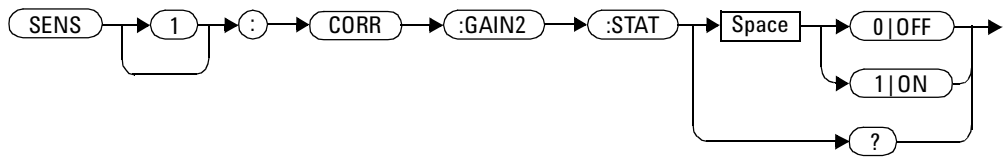
```
[SENSE[1]]:CORRection:GAIN2:STATE <boolean>
```

```
[SENSE[1]]:CORRection:GAIN2[:INPut][:MAGNitude] <numeric_value>
```

[SENSe[1]]:CORRection:GAIN2:STATe <boolean>

This command is used to enable/disable a channel offset for the power sensor setup. The [SENSe[1]]:CORRection:GAIN2[:INPut] [:MAGNitude] command is used to enter the loss/gain value.

Syntax



Example

```
CORR:GAIN2:STAT ON
```

This command enables a channel offset.

Reset Condition

On reset, channel offsets are disabled.

Query

```
[SENSe[1]]|SENSe2:CORRection:GAIN2:STATe?
```

The query enters 1 or 0 into the output buffer indicating the status of the channel offsets.

- 1 is returned if a channel offset is enabled
- 0 is returned if a channel offset is disabled

Query Example

```
CORR:GAIN2:STAT?
```

This command queries whether or not there is a channel offset applied.

Error Messages

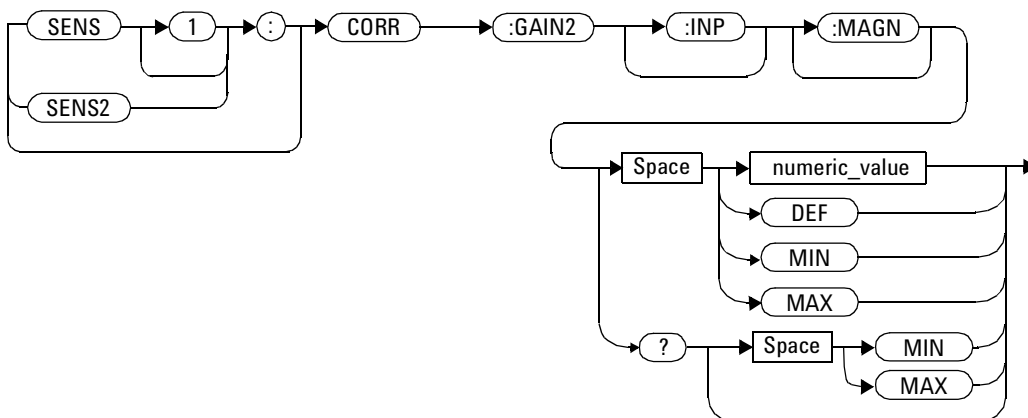
If [SENSe[1]]|SENSe2:CORRection:GAIN2:STATe is set to ON while [SENSe[1]]|SENSe2: is set to 200, the error -221, “Settings Conflict” occurs.

[SENSe[1]] | SENSE2:CORRection:GAIN2[:INPut] [:MAGNitude] <numeric_value>

This command is used to enter a channel offset value for the power meter setup, for example cable loss. The power meter then corrects every measurement by this factor to compensate for the gain/loss.

Entering a value for GAIN2 using this command automatically turns the [SENSe[1]] | SENSE2:CORRection:GAIN2:STATe command to ON.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value: <ul style="list-style-type: none"> • DEF: the default is 0.00 dB • MIN: -100 dB • MAX: +100 dB 	-100 to +100 dB DEF MIN MAX

Example

```
CORR:GAIN2 50
```

This command sets a channel offset of 50 dB.

Reset Condition

On reset, GAIN2 is set to 0.00 dB.

Query

```
[SENSe[1]]|SENSe2:CORRection:GAIN2[:INPut][:MAGNitude]?  
[MIN|MAX]
```

The query returns the current setting of the channel offset or the values associated with MIN and MAX.

Query Example

`CORR:GAIN2?`

This command queries the current setting of the channel offset.

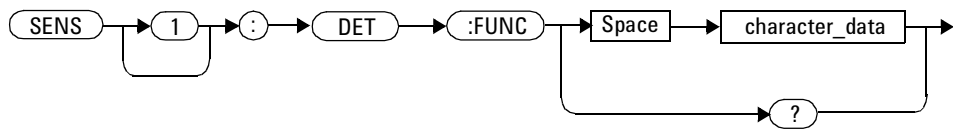
Error Messages

- If a loss/gain correction value is entered using `[SENSE[1]]|SENSE2:CORREction:GAIN2[:INPut][:MAGNitude]` while `[SENSE[1]]|SENSE2:` is set to 200, the error -221, “Settings Conflict” occurs. However, the correction value is set but the `[SENSE[1]]|SENSE2:CORREction:GAIN2:STATe` command is not automatically set ON.

SENSe[1]:DETECTOR:FUNCTION <character_data>

This command sets the measurement mode for U2000 Series USB power sensors.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Defines the measurement mode: <ul style="list-style-type: none"> AVERAge: sets the sensor to average only mode. 	AVERAge ¹

¹ When measurement mode is set to average:

- If TRIG:SOUR is set to INT1, or EXT, it is set automatically to IMM.
- INIT:CONT is set automatically to ON.
- SENS:AVER2:STAT is set automatically to OFF.
- CALC:FEED is set automatically to "POW:AVG" for all CALC blocks using the current measurement in their CALC:MATH:EXPR.

Example

```
SENS1:DET:FUNC AVER
```

This command sets the sensor to Average mode.

Reset Condition

On reset, the mode is set to AVERage.

Query

```
SENSe[1]:DETECTOR:FUNCTION?
```

The query returns the current sensor mode setting.

Query Example

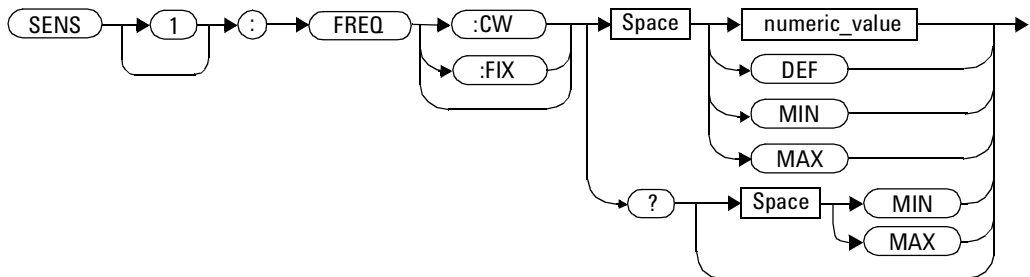
```
SENS:DET:FUNC?
```

This command queries the current sensor mode setting.

SENSe[1]:FREQuency[:CW | :FIXed] <numeric_value>

This command is used to enter a frequency. The appropriate frequency dependent offset corrections are applied for the frequency selected, dependant on the frequency dependent offset data stored in the sensor's non-volatile memory.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value for the frequency: <ul style="list-style-type: none"> • DEF: the default value is 50 MHz • MIN: 1 kHz • MAX: 1000.0 GHz The default units are Hz.	1 kHz to 1000.0 GHz ¹ DEF MIN MAX

¹ The following measurement units can be used:

- Hz
- kHz (10^3)
- MHz (10^6)
- GHz (10^9)

Example

```
FREQ 500kHz
```

This command enters a sensor frequency of 500 kHz.

Reset Condition

On reset, the frequency is set to 50 MHz (DEF).

Query

```
SENSe[1]:FREQuency[:CW|:FIXed]? [MIN|MAX]
```

The query returns the current frequency setting or the values associated with MIN and MAX. The units in which the results are returned are Hz.

Query Example

```
SENS1:FREQ?
```

This command queries the sensor frequency setting.

SENSe[1]:MRATe <character_data>

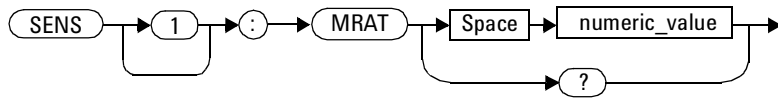
This command sets the measurement on the power sensor.

When a channel is set to FAST, the following couplings occur:

Command	Status
SENSe[1]:AVERAge:STATe	OFF ¹
SENSe[1]:CORRection:GAIN2:STATe	OFF ¹
CALCulate1:MATH:EXPRession	"(SENSe1)"

¹ This change only occurs on the measurement specified in the SENSe:MRATe command. When the sensor is changed from FAST to NORMAl or DOUBle, the settings that were in place when FAST was entered are restored.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	A numeric value for the measurement : <ul style="list-style-type: none"> • NORMAl: 20 readings/second • DOUBle: 40 readings/second • FAST : up to 1000 readings/second The default is NORMAl.	NORMAl ¹ DOUBle ¹ FAST

¹ When a channel is set to NORMAl or DOUBle, TRIG:COUNT is set automatically to 1.

Example

```
MRAT DOUBle
```

This command sets the sensor to 40 readings/second.

Reset Condition

On reset, the is set to NORMAl.

Query

```
SENSe[1]:MRAT?
```

The query returns the current setting, either NORMAL, DOUBLE or FAST.

Query Example

```
MRAT?
```

This command queries the current sensor setting.

Error Messages

- If <character_data> is not set to NORMAL, DOUBLE or FAST, error -224 “Illegal parameter value” occurs.

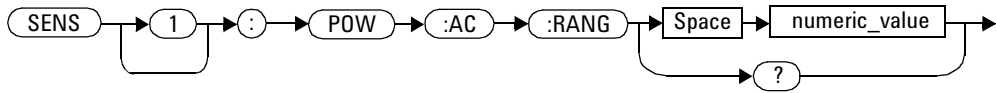
SENSe[1]:POWer:AC:RANGe <numeric_value>

Its purpose is to select one of two power ranges.

- If 0 is selected, the power sensor's lower range is selected
- If 1 is selected, the power sensor's upper range is selected

Setting a range with this command automatically switches SENSE[1]:POWer:AC:RANGe:AUTO to OFF.

Syntax



Example

```
POW:AC:RANG 0
```

This command sets the power sensor to its lower range.

Reset Condition

On reset, the upper range is selected.

Query

```
SENSe[1]:POWer:AC:RANGe?
```

The query enters a 1 or 0 into the output buffer indicating the status of the power sensor's range.

- 1 is returned when the upper range is selected
- 0 is returned when the lower range is selected

Query Example

POW:AC:RANG?

This command queries the current setting of the power sensor range.

Error Messages

This command is used with the U2000 Series USB power sensors. If one is not connected the error -241, “Hardware missing” occurs.

NOTE

For U2000 Series USB power sensors, the auto ranging feature will be disabled when trigger modes are selected.

SENSe[1]:POWer:AC:RANGe:AUTO <boolean>

Its purpose is to enable and disable autoranging. When autoranging is ON, the power sensor selects the best measuring range for the measurement. When autoranging is set to OFF, the power sensor remains in the currently set range.

The `SENSe[1]:POWer:AC:RANGe` command disables autoranging.

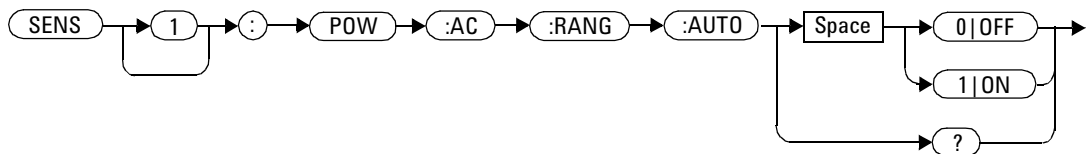
If `INITiate:CONTinuous` is set to ON and `TRIGger:SOURce` is set to `IMMediate`, the range tracks the input power if `SENSe[1]:POWer:AC:RANGe:AUTO` is ON.

If the power sensor is not making measurements then autoranging only occurs when the power sensor is triggered.

NOTE

For U2000 Series USB power sensors, only HIGH and LOW ranges are available in Triggered modes.

Syntax



Example

```
POW:AC:RANG:AUTO 0
```

This command disables autoranging.

Reset Condition

On reset, autoranging is enabled.

Query

```
SENSE[1]:POWER:AC:RANGE:AUTO?
```

The query enters a 1 or 0 into the output buffer indicating the status of autoranging.

- 1 is returned when autoranging is enabled
- 0 is returned when autoranging is disabled

Query Example

```
POW:AC:RANG:AUTO?
```

This command queries whether auto ranging is on or off.

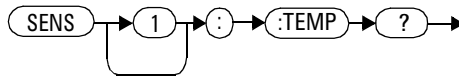
Error Messages

- If this command is set to ON when U2000 Series USB Power Sensor is in Triggered modes, the error -221, “Setting conflicts” occurs.

SENSe[1]:TEMPerature?

This this command to returns the power sensor's temperature in degrees Celsius.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	A numeric value defining sensor's temperature in degrees Celsius.	-50 to 100

Example

```
SENS1:TEMP?
```

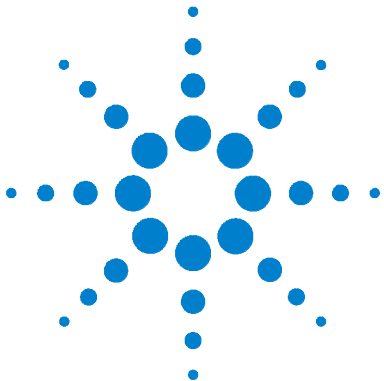
This command returns the current sensor temperature.

Reset Condition

On reset, this parameter is not affected.

Error Messages

- If a U2000 Series USB Power Sensor is not connected, error -241, “Hardware missing” occurs.



8 SERvice Subsystem

SERvice Subsystem	200
SERvice:BIST:TRIGger:LEVel:STATe?	202
SERvice:OPTion <character_data>	203
SERvice:SENSor[1]:CDATe <"date">	204
SERvice:SENSor[1]:CPLace <"place">	205
SERvice:SENSor[1]:FREQuency:MAXimum?	207
SERvice:SENSor[1]:FREQuency:MINimum?	208
SERvice:SENSor[1]:POWer:AVERage:MAXimum?	209
SERvice:SENSor[1]:POWer:USABle:MAXimum?	210
SERvice:SENSor[1]:POWer:USABle:MINimum?	211
SERvice:SENSor[1]:RADc?	212
SERvice:SENSor[1]:SNUMber <"serial_number">	213
SERvice:SENSor[1]:TNUMber <"tracking_number">	214
SERvice:SENSor[1]:TYPE?	216
SERvice:SNUMber <character_data>	217
SERvice:VERSion:PROcEссор <character_data>	218
SERvice:VERSion:SYSTem <character_data>	220

This chapter explains how the `SERvice` command subsystem is used to obtain and set information useful for servicing the power sensor.



SERVICE Subsystem

The SERVICE command subsystem is used to load information such as the power sensor processor board revision version and obtain information such as the serial number of the current sensor(s) being used.

Keyword	Parameter Form	Notes	Page
SERVICE			
:BIST			
:TRIGger			
:LEVel			
:STATe?		[query only]	page 202
:OPTion	<character_data>		page 203
:SENSor[1]			
:CDATe	<"date">	[query only]	page 204
:CPLace	<"place">	[query only]	page 205
:FREQuency			
:MAXimum?		[query only]	page 207
:MINimum?		[query only]	page 208
:POWer			
:AVERage			
:MAXimum?		[query only]	page 209
:USABle			
:MAXimum?		[query only]	page 210
:MINimum?		[query only]	page 211
:RADC?		[query only]	page 212
:SNUMber	<"serial_number">	[query only]	page 213
:TNUMBER?	<"tracking_number">	[query only]	page 214
:TYPE?		[query only]	page 216
:SNUMber	<character_data>		page 217
:VERSion			
:PROCCessor	<character_data>		page 218

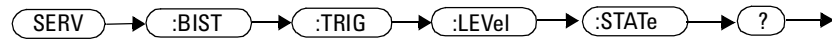
Keyword	Parameter Form	Notes	Page
:SYSTEM	<character_data>		page 220

SERVICE:BIST:TRIGGER:LEVEL:STATE?

This command queries trigger level.

- 1 is returned when the external trigger-in is high.
- 0 is returned when the external trigger-in is low.

Syntax



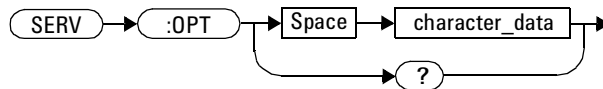
Example

SERV:BIST:TRIG:LEV:STAT? *This command queries trigger level.*

SERvice:OPTion <character_data>

This command loads the power sensor memory with the options fitted. The query form of the command can be used to determine which options are fitted to the unit.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the option number in a comma separated list. A maximum of 30 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9 _ (underscore)

Example

```
SERV:OPT "003"
```

This command loads the power sensor memory with 003 indicating that the unit is a rear panel option.

Query

```
SERvice:OPTion?
```

The query returns the current option string.

SERVICE:SENSor[1]:CDATe <"date">

This command is used to enter the calibration date.

Syntax



Example

```
SERV:SENS1:CDAT
"2006,09,21"
```

This command enters the calibration date of the U2000 Series USB Power Sensor as 21st September 2006.

Query

```
SERVICE:SENSor[1]:CDATe?
```

This query returns the calibration date in U2000 Series USB power sensors. Calibration date information is stored in the sensor's non-volatile memory.

Query example

```
SERV:SENS1:CDATe?
```

This query returns the calibration date of the U2000 Series USB Power Sensor.

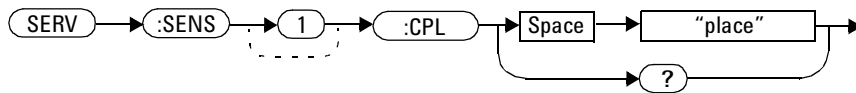
NOTE

If calibration date is not pre-programmed, "NONE" string is returned.

SERvice:SENSor[1]:CPLace <"place">

This query is used to enter the calibration place.

Syntax



Example

```
SERvice:SENS1:CPL
"Agil-Penang"
```

This command enters the place of calibration of the U2000 Series USB power sensor as Agilent Penang.

Query

```
SERvice:SENSor[1]:CPLace?
```

This query returns the calibration place in U2000 Series USB power sensors. Calibration place information is stored in the sensor's non-volatile memory.

Query example

SERV:SENS1:CPL?

This query returns the place of calibration of the U2000 Series USB Power Sensor.

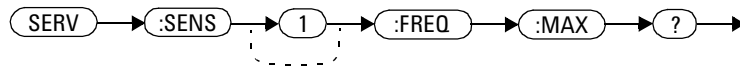
NOTE

If calibration date is not pre-programmed, "NONE" string is returned.

SERvice:SENSor[1]:FREQuency:MAXimum?

This query returns the maximum frequency that can be measured in MHz unit by the currently connected sensor. It is applicable to U2000 Series USB power sensors only. Maximum frequency information is stored in the sensor's non-volatile memory.

Syntax



Example

SERV:SENS1:FREQ:MAX?

This query returns the maximum frequency that can be measured by the U2000 Series USB Power Sensor currently.

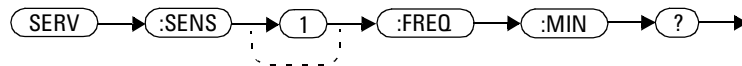
Error Messages

- If the sensor, currently connected, does not contain the necessary information in non-volatile memory, error -241 “Hardware missing” occurs.

SERVICE:SENSOR[1]:FREQUENCY:MINIMUM?

This query returns the minimum frequency that can be measured in MHz unit by the currently connected sensor. It is applicable to U2000 Series USB power sensors only. Minimum frequency information is stored in the sensor's non-volatile memory.

Syntax



Example

```
SERV:SENS1:FREQ:MIN?
```

This query returns the minimum frequency that can be measured by the U2000 Series USB Power Sensor currently.

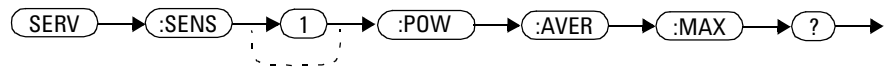
Error Messages

- If the U2000 Series USB Power Sensor currently connected does not contain the necessary information in non-volatile memory, error -241 “Hardware missing” occurs.

SERvice:SENSor[1]:POWer:AVERage:MAXimum?

This query returns the maximum average power that can be measured in dBm unit by the currently connected sensor. Maximum average power information is stored in the sensor's non-volatile memory.

Syntax



Example

SERV:SENS:POW:AVER:MAX?

This query returns the maximum average power that can be measured by the U2000 Series USB Power Sensor.

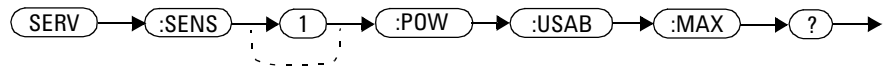
Error Messages

- If the U2000 Series USB Power Sensor currently connected does not contain the necessary information in non-volatile memory, error -241 “Hardware missing” occurs.

SERVICE:SENSOR[1]:POWER:USABLE:MAXimum?

This query returns the maximum power that can be accurately measured in dBm unit by the currently connected sensor. Maximum power information is stored in the sensor's non-volatile memory.

Syntax



Example

`SERV:SENS1:POW:USAB:MAX?` *This query returns the maximum power that can be accurately measured by the U2000 Series USB Power Sensor.*

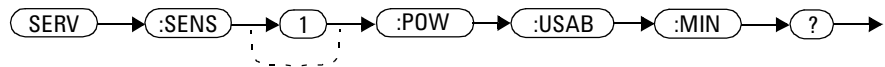
Error Messages

- If the U2000 Series USB Power Sensor currently connected does not contain the necessary information in non-volatile memory, error -241 “Hardware missing” occurs.

SERvice:SENSor[1]:POWer:USABle:MINimum?

This query returns the minimum power that can be accurately measured in dBm unit by the currently connected sensor. Maximum power information is stored in the sensor's non-volatile memory.

Syntax



Example

SERV:SENS:POW:USAB:MIN?

This query returns the minimum power that can be accurately measured by the U2000 Series USB Power Sensor.

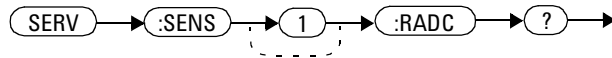
Error Messages

- If the U2000 Series USB Power Sensor currently connected does not contain the necessary information in non-volatile memory, error -241 “Hardware missing” occurs.

SERVICE:SENSOR[1]:RADC?

This query returns a new raw uncorrected measurement in volts, as a 32 bit signed integer.

Syntax



Example

SERV:SENS1:RADC?

This query returns a new raw uncorrected measurement for the sensor.

NOTE

Returned raw measurement is in float type unit.

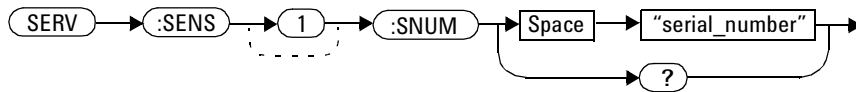
Error Messages

- If INIT:CONT is set to ON, error -221 “Settings Conflict” occurs.

SERvice:SENSor[1]:SNUMber <"serial_number">

This command is used to enter the serial number of U2000 Series USB power sensors.

Syntax



Example

`SERV:SENS1:SNUM MY12345678` *This command enters the serial number and the changes of serial number will take effect after power recycle.*

Query

`SERvice:SENSor[1]:SNUMber?`

This query returns the serial number for U2000 Series USB power sensors. Serial number information is stored in the sensor's non-volatile memory.

Query Example

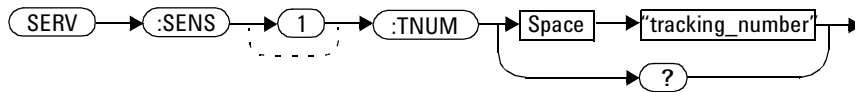
`SERV:SENS1:SNUM?`

This query returns the serial number of the U2000 Series USB Power Sensor.

SERVICE:SENSOr[1]:TNUMBER <"tracking_number">

This command is used to enter the tracking number of U2000 Series USB power sensors.

Syntax



Example

```
SERV:SENS1:TNUM
"PEN12345"
```

This command enters the tracking number of PEN12345.

Query

```
SERVICE:SENSOr[1]:TNUMBER?
```

This query returns the tracking number for U2000 Series USB power sensors. Tracking number information is stored in the sensor's non-volatile memory.

Query Example

```
SERV:SENS1:TNUM?
```

This query returns the tracking number of the U2000 Series USB Power Sensor.

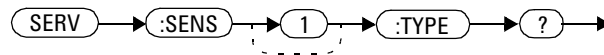
NOTE

If tracking number is not pre-programmed, "NONE" string is returned.

SERVICE:SENSOR[1]:TYPE?

This query identifies the sensor type, either U2000A, U2001A, U2002A, U2004A, U2001H, U2001B, U2000H, U2000B, U2002H and U2002B.

Syntax



Example

SERV:SENS1:TYPE?

This query returns either, U2000A, U2001A, U2002A, U2004A, U2001H, U2001B, U2000H, U2000B, U2002H and U2002B.

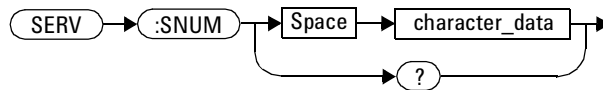
Error Messages

- If no sensor is connected, error -241, “Hardware missing” occurs.

SERVice:SNUMber <character_data>

This command loads the power sensor with a serial number in the form GB12345678 or US12345678.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the power sensor serial number in the form GB12345678 or US12345678. A maximum of 30 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9

Example

```
SERV:SNUM GB12345678
```

This command loads the power sensor with the serial number GB12345678.

Query

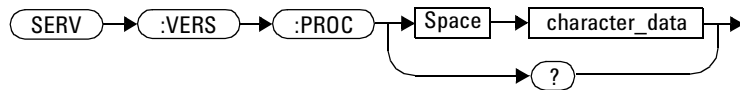
```
SERVice:SNUMber?
```

The query returns the power sensor serial number in the form GB12345678 or US12345678.

SERVICE:VERSION:PROCESSOR <character_data>

This command loads the power sensor with the processor board revision version.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the processor board revision version. A maximum of 20 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9 _ (underscore)

Example

```
SERV:VERS:PROC "C"
```

This command loads the power sensor with processor board revision version C.

Query

```
SERVICE:VERSION:PROCESSOR?
```

The query returns the current processor board revision version.

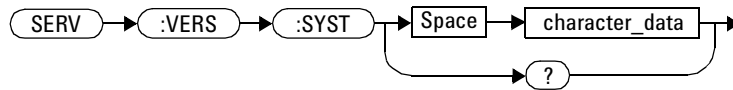
NOTE

If processor board revision is not pre-programmed, "NONE" string is returned.

SERVICE:VERSION:SYSTEM <character_data>

This command loads the power sensor with the system version number.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	Details the system version number. A maximum of 20 characters can be used.	A to Z (uppercase) a to z (lowercase) 0 - 9 _ (underscore)

Example

```
SERV:VERS:SYST "1"
```

This command loads the power sensor with system version number 1.

Query

```
SERVICE:VERSION:SYSTEM?
```

The query returns the current power sensor system version number.

NOTE

If system version number is not pre-programmed, "NONE" string is returned.



9 STATus Subsystem

STATus Subsystem	224
Status Register Set Commands	226
Device Status Register Sets	231
Operation Register Sets	232
STATus:OPERation	233
STATus:OPERation:CALibrating[:SUMMARY]	234
STATus:OPERation:LLFail[:SUMMARY]	235
STATus:OPERation:MEASuring[:SUMMARY]	236
STATus:OPERation:SENSe[:SUMMARY]	237
STATus:OPERation:TRIGger[:SUMMARY]	238
STATus:OPERation:ULFail[:SUMMARY]	239
STATus:PRESet	240
Questionable Register Sets	241
STATus:QUEStionable	242
STATus:QUEStionable:CALibration[:SUMMARY]	243
STATus:QUEStionable:POWer[:SUMMARY]	244

This chapter explains how the *STATus* command subsystem enables you to examine the status of the power sensor by monitoring the “Device Status Register”, “Operation Status Register” and the “Questionable Status Register”.



STATUS Subsystem

The STATUS command subsystem enables you to examine the status of the power sensor by monitoring the following status registers:

- Device status register
- Operation status register
- Questionable status register

The contents of these and other registers in the power sensor are determined by one or more status registers.

Table 9-1 summarizes the effects of various commands and events on these status registers:

Table 9-1 Commands and events affecting Status Register

Status Register	*RST	*CLS	Power On	STATUS: PRESet
SCPI Transition Filters (NTR and PTR registers)	none	none	preset	preset
SCPI Enable Registers	none	none	preset	preset
SCPI Event Registers	none	clear	clear	none
SCPI Error/Event Queue enable	none	none	preset	preset
SCPI Error/Event Queue	none	clear	clear	none
IEEE488.2 Registers ESE SRE	none	none	clear	none
IEEE488.2 Registers SESR STB	none	clear	clear	none

The contents of the status registers are examined using the following status register set commands:

```
:CONDition?
:ENABle <NRf>|<non-decimal numeric>
[:EVENT?]
:NTRansition <NRf>|<non-decimal numeric>
:PTRansition <NRf>|<non-decimal numeric>
```

Each of these can be used to examine any of the following eleven status registers:

STATUS:DEVICE (page 231)
 STATUS:OPERATION (page 233)
 STATUS:OPERATION:CALIBRATING[:SUMMARY] (page 234)
 STATUS:OPERATION:LLFAIL[:SUMMARY] (page 235)
 STATUS:OPERATION:MEASURING[:SUMMARY] (page 236)
 STATUS:OPERATION:SENSE[:SUMMARY] (page 237)
 STATUS:OPERATION:TRIGGER[:SUMMARY] (page 238)
 STATUS:OPERATION:ULFAIL[:SUMMARY] (page 239)
 STATUS:QUESTIONABLE (page 240)
 STATUS:QUESTIONABLE:CALIBRATION[:SUMMARY] (page 243)
 STATUS:QUESTIONABLE:POWER[:SUMMARY] (page 244)

Examples

- To use the :CONDITION? command to examine the STATUS:DEVICE register:

```
STATUS:DEVICE:CONDITION?
```

- To use the :NTRANSITION command to examine the STATUS:OPERATION:SENSE[:SUMMARY] register:

```
STATUS:OPERATION:SENSE[:SUMMARY]:NTRANSITION
```

This chapter describes the status register set commands and the status registers which they are used to examine.

Status Register Set Commands

This section describes the five status register set commands. Each can be used to examine all of the eleven status registers listed on [page 225](#).

To apply a command to a specific register, prefix the command with the name of the appropriate register. For example, to apply the :ENABLE command to the STATUS:QUESTIONABLE register, use the following command:

```
STATUS:QUESTIONABLE:ENABLE
```

The Status Register Set commands detailed in this section are:

Keyword	Parameter Form	Notes	Page
:CONDition?		[query only]	page 226
:ENABLE	<NRf> <non-decimal numeric>		page 227
[:EVENT?]		[query only]	page 227
:NTRansition	<NRf> <non-decimal numeric>		page 228
:PTRansition	<NRf> <non-decimal numeric>		page 229

:CONDition?

This query returns a 16 bit decimal-weighted number representing the bits set in the Condition Register of the SCPI Register Set you require to control. The format of the return is <NR1> in the range of 0 to 32767 ($2^{15}-1$). The contents of the Condition Register remain unchanged after it is read.

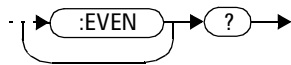
Syntax

```
--> :COND --> ? -->
```

[:EVENTt]?

This query returns a 16 bit decimal-weighted number representing the bits set in the Event Register of the SCPI Register Set you require to control. The format of the return is <NR1> in the range of 0 to 32767 ($2^{15}-1$). This query clears all bits in the register to 0.

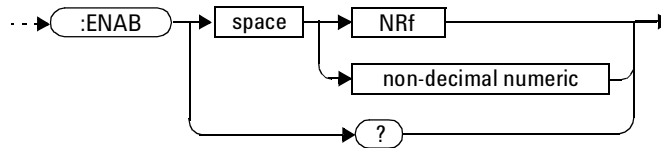
Syntax



:ENABLE <NRf> | <non-decimal numeric>

This command sets the Enable Register of the particular SCPI Register Set you require to control. The parameter value, when rounded to an integer and expressed in base 2 has its first 15 bits written into the Enable Register of the SCPI Register Set concerned. The last bit (bit 15) is always set to 0.

Syntax



Parameters

Type	Description	Range of Values
NRf	The value used to set the Enable Register.	0 to $2^{16}-1$
non-decimal numeric		

Query

:ENABle?

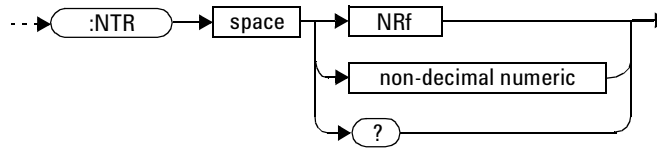
The query returns a 15 bit decimal-weighted number representing the contents of the Enable Register of the SCPI Register Set being queried.

The format of the return is <NR1> in the range of 0 to 32767 ($2^{15}-1$).

:NTRansition <NRf> | <non-decimal numeric>

This command sets the Negative Transition Register of the SCPI Register Set you require to control. The parameter value, when rounded to an integer and expressed in base 2 has its first 15 bits written into the Negative Transition Register of the SCPI Register Set concerned. The last bit (bit 15) is always set to 0.

Syntax



Parameters

Type	Description	Range of Values
NRf	The value used to set the NTR Register.	0 to $2^{16}-1$
non-decimal numeric		

Query

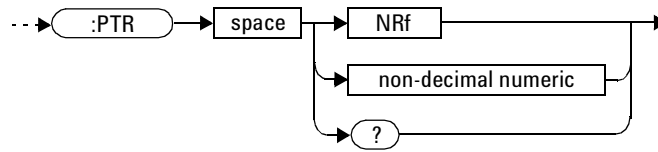
:NTRansition?

The query returns a 15 bit decimal-weighted number representing the contents of the Negative Transition Register of the SCPI register set being queried. The format of the return is <NR1> in the range of 0 to 32767 ($2^{15}-1$).

:PTRansition <NRf> | <non-decimal numeric>

This command is used to set the Positive Transition Register of the SCPI Register Set you require to control. The first 15 bits of the input parameter are written into the Positive Transition Register of the SCPI Register Set concerned. The last bit (bit 15) is always set to 0.

Syntax



Parameters

Type	Description	Range of Values
NRf	The value used to set the PTR Register.	0 to $2^{16}-1$
non-decimal numeric		

Query

`:PTRansition?`

The query returns a 15 bit decimal-weighted number representing the contents of the Positive Transition Register of the SCPI register set being queried. The format of the return is <NR1> in the range of 0 to 32767 ($2^{15}-1$).

Device Status Register Sets

The status registers contain information which give device status information. The contents of the individual registers of these register sets may be accessed by appending the commands listed in “[Status Register Set Commands](#)” on page 226.

The following command descriptions detail the SCPI register you require to control but do not detail the register set commands.

The one device status register set is:

STATUS:DEVICE:

The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Not used
2	4	Not used
3	8	Power sensor error
4	16	Not used
7-15	-	Not used
14	16384	Not used
15	-	Bit 15 always 0

The power sensor error bits (3) are set to:

- 1, if the U2000 Series USB power sensor’s non-volatile memory has failed.
- 0, for every other condition.

Operation Register Sets

The following registers contain information which is part of the power sensor's normal operation. The contents of the individual registers of these register sets may be accessed by appending the commands listed in [“Status Register Set Commands”](#) on page 226.

The following command descriptions detail the SCPI register you require to control but do not detail the Register Set commands.

The seven Operation Register Sets are:

```
STATUS:OPERation
STATUS:OPERation:CALibrating[:SUMMARY]
STATUS:OPERation:LLFail[:SUMMARY]
STATUS:OPERation:MEASuring[:SUMMARY]
STATUS:OPERation:SENSe[:SUMMARY]
STATUS:OPERation:TRIGger[:SUMMARY]
STATUS:OPERation:ULFail[:SUMMARY]
```

Further information on these register sets is provided on the following pages.

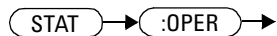
STATus:OPERation

The operation status register set contains conditions which are a part of the operation of the power sensor as a whole.

The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	1	CALibrating Summary
1 - 3	-	Not used
4	16	MEASuring Summary
5	32	Waiting for TRIGger Summary
6 - 9	-	Not used
10	1024	SENSE Summary
11	2048	Lower Limit Fail Summary
12	4096	Upper Limit Fail Summary
13 to 15	-	Not used (bit 15 always 0)

Syntax



STATUS:OPERation:CALibrating[:SUMMARY]

The operation status calibrating summary register set contains information on the calibrating status of the power sensor.

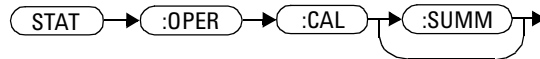
The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	CALibrating Status
2	4	Not used
3-15	-	Not used

These bits are set at the beginning of zeroing (CALibration:ZERO:AUTO ONCE) and at the beginning of calibration (CALibration:AUTO ONCE). Also for the compound command/query CALibration[:ALL]?, this bit is set at the beginning of the calibration sequence.

These bits are cleared at the end of zeroing or calibration.

Syntax



STATus:OPERation:LLFail[:SUMM]ary]

The operation status lower limit fail summary register set contains information on the lower limit fail status of the power sensor.

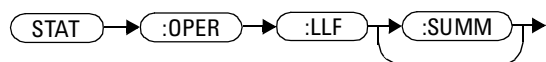
The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Current window LLFail Status
2	4	Not used
3	8	Not used
4	16	Not used
5	32	Not used
6	64	Not used
7-15	-	Not used

The appropriate bits are set if a channel lower limit test fails or a window lower limit test fails.

These bits are cleared if a measurement is made and the test is enabled and passes.

Syntax



STATus:OPERation:MEASuring[:SUMMary]

The operation status measuring summary register set contains information on the measuring status of the power sensor.

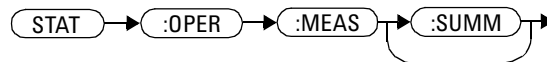
The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Current window MEASuring Status
2	4	Not used
3-15	-	Not used

These bits are set when the power sensor is taking a measurement.

These bits are cleared when the measurement is finished.

Syntax



STATus:OPERation:SENSe[:SUMM]ary

The operation status sense summary register set contains information on the status of the power sensor.

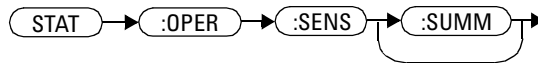
The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Current window SENSE Status
2	4	Not used
3-15	-	Not used

These bits are set when the power sensor is reading data from the power sensor non-volatile memory.

These bits are cleared when the power sensor is not reading data from the power sensor non-volatile memory.

Syntax



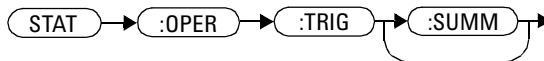
STATus:OPERation:TRIGger[:SUMMary]

The operation status trigger summary register set contains information on the trigger status of the power sensor.

The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Current window TRIGger Status
2	4	Not used
3-15	-	Not used

Syntax



STATus:OPERation:ULFail[:SUMM]ary

The operation status upper limit fail summary register set contains information on the upper limit fail status of the power sensor.

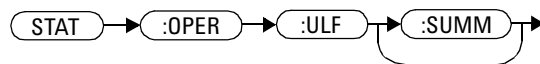
The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Current window ULFail Status
2	4	Not used
3	8	Not used
4	16	Not used
5	32	Not used
6	64	Not used
7-15	-	Not used

The appropriate bits are set if a channel upper limit test fails or a window upper limit test fails.

These bits are cleared if a measurement is made and the test is enabled and passes.

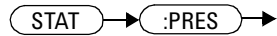
Syntax



STATUS:PRESet

PRESet sets a number of the status registers to their preset values as shown below—all other registers are unaffected. Bit 15 is always 0.

Syntax



Register	Filter/Enable	PRESet Value
OPERational	ENABLE	all zeros
	PTR	all ones
	NTR	all zeros
QUESTionable	ENABLE	all zeros
	PTR	all ones
	NTR	all zeros
All Others	ENABLE	all ones
	PTR	all ones
	NTR	all zeros

Questionable Register Sets

The questionable register sets contain information which gives an indication of the quality of the data produced by the power sensor. The contents of the individual registers in these register sets may be accessed by appending the commands listed in “[Status Register Set Commands](#)” on page 226.

The following command descriptions detail the SCPI register you require to control but do not detail the register set commands.

The three questionable register sets are:

```
STaTus:QUEStionable
```

```
STaTus:QUEStionable:CALibration[:SUMMary]
```

```
STaTus:QUEStionable:POWer[:SUMMary]
```

STATus:QUEStionable

The questionable register set contains bits that indicate the quality of various aspects of signals processed by the power sensor.

The following bits in these registers are used by the power sensor:

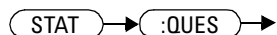
Bit Number	Decimal Weight	Definition
0 to 2	-	Not used
3	8	POWer Summary
4 to 7	-	Not used
8	256	CALibration Summary
9	512	Power On Self Test
10 to 15	-	Not Used (bit 15 always 0)

Bit 3 is set by the logical OR outputs of the `STATus:QUEStionable:POWer:SUMMary` register set.

Bit 8 is set by the logical OR outputs of the `STATus:QUEStionable:CALibration:SUMMary` register set.

Bit 9 is set if power-on self-test fails, and cleared if it passes.

Syntax



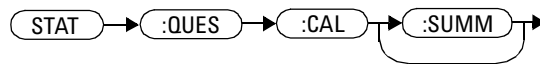
STATus:QUEStionable:CALibration[:SUMM]ary

The questionable calibration summary register set contains bits which give an indication of the quality of the data produced by the power sensor due to its calibration status.

The following bits in these registers are used by the power sensor:

Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Summary of current window CALibration
2	4	Not used
3-15	-	Not used

Syntax



STATUS:QUESTIONABLE:POWER[:SUMMARY]

The questionable power summary register set contain bits that indicate the quality of the power data being acquired by the power sensor.

The following bits in these registers shall be used by the power sensor:

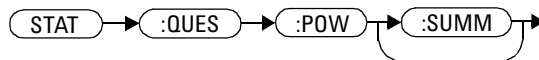
Bit Number	Decimal Weight	Definition
0	-	Not used
1	2	Power
2	4	Not used
3	8	Not used
4	16	Not used
5	32	Not used
6	64	Not used
7	128	Not used
8	256	Not used

Bit 1 is set when any of the following errors occur:

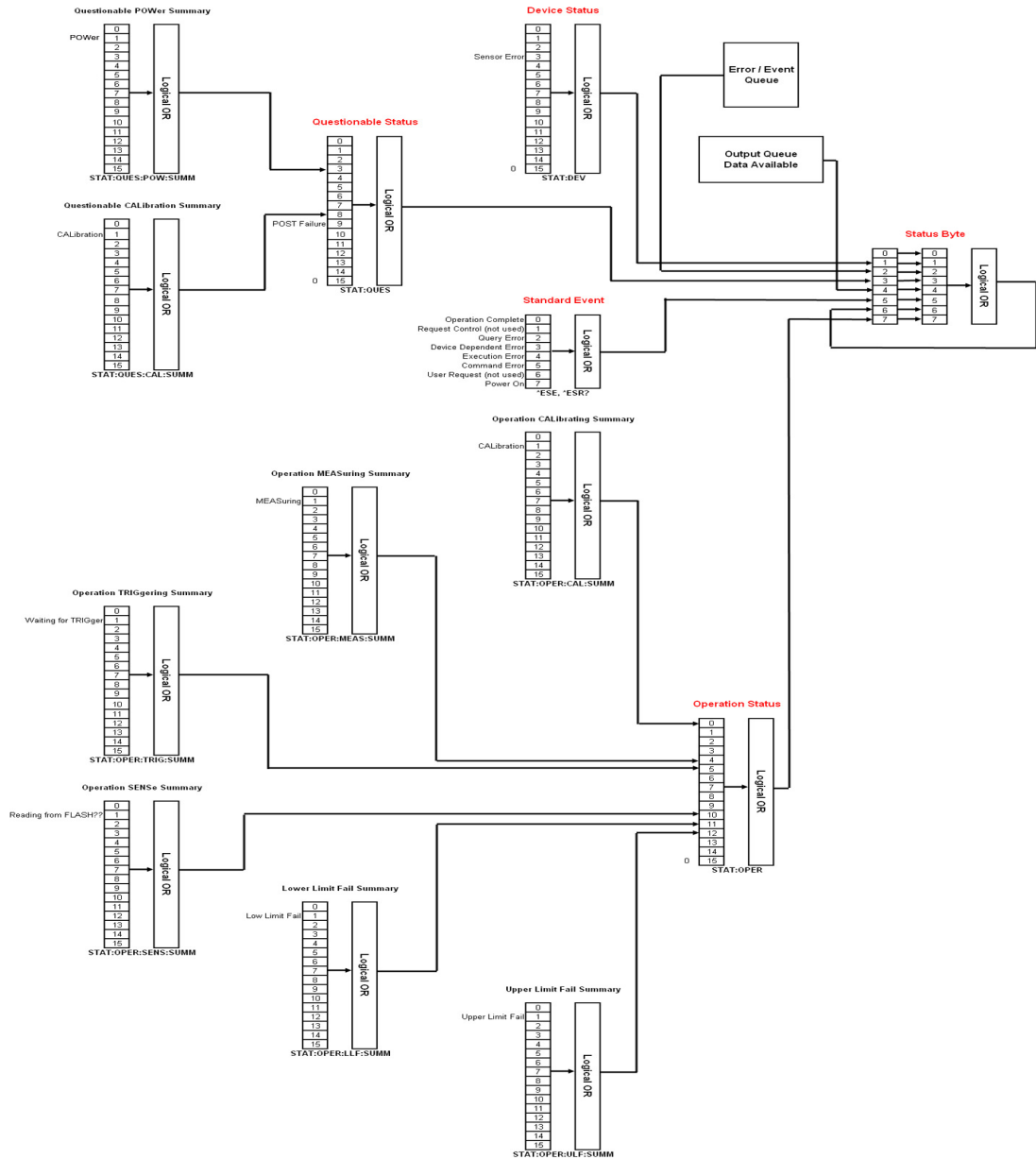
- Error -231, “Data questionable;Input Overload”

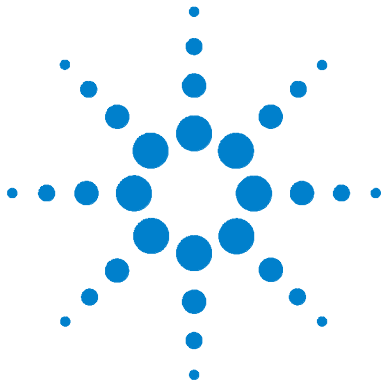
This bit is cleared when no errors or events are detected by the power sensor during a measurement covering the causes given for it to set.

Syntax



Status Block Diagram





10 SYSTEM Subsystem

SYSTEM Subsystem	248
SYSTEM:ERRor?	249
SYSTEM:HELP:HEADers?	256
SYSTEM:PRESet <character_data>	257
SYSTEM:VERSion?	260

This chapter explains how to use the `SYSTEM` command subsystem to return error numbers and messages from the power sensor, preset the power sensor, and query the SCPI version.



SYSTEM Subsystem

The SYSTEM command subsystem is used to:

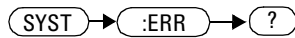
- Return error numbers and messages from the power sensor
- Preset the power sensor
- Query the SCPI version

Keyword	Parameter Form	Notes	Page
SYSTEM			
:ERRor			page 249
:HELP			
:HEADers?		[query only]	page 256
:PRESet			page 257
:VERSion?		[query only]	page 260

SYSTem:ERRor?

This query returns error numbers and messages from the power sensor’s error queue. When an error is generated by the power sensor, it stores an error number and corresponding message in the error queue. One error is removed from the error queue each time this command is executed. The errors are cleared in the order of first-in first-out, this is the oldest errors are cleared first. To clear all the errors from the error queue, execute *CLS command. When the error queue is empty, subsequent SYSTem:ERRor? queries return a +0, “No error” message. The error queue has a maximum capacity of 50 errors.

Syntax

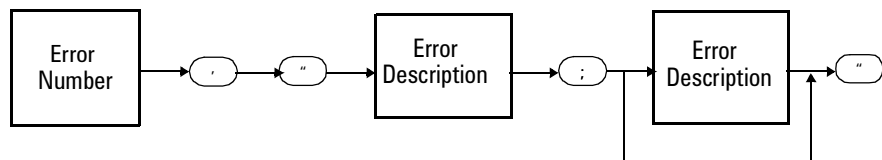


Query Example

SYST:ERR?

This command queries the system error.

Error queue messages have the following format:



Error Queue Message

For example, -330, “Self-test Failed;Battery Fault”.

Errors are retrieved in a first in first out (FIFO) order. If more than 30 errors occur, the error queue overflows and the last error in the queue is replaced with the error -350, “Queue Overflow”. Any time the queue is overflows the most recent error is discarded.

Example

SYST:ERR?

This command queries the oldest error message stored in the power sensor's error queue.

Reset Condition

On reset, the error queue is unaffected.

Error Messages

- If the error queue overflows, the last error is replaced with -350, “Queue overflow”. No additional errors are accepted by the queue until space becomes available.

Error Message List

-101	<p>Invalid character</p> <p>An invalid character was found in the command string. You may have inserted a character such as #, \$, or % in the command header or within a parameter.</p> <p>For example, LIM:LOW 0#.</p>
-102	<p>Syntax error</p> <p>Invalid syntax was found in the command string.</p> <p>For example, LIM:CLE:AUTO, 1 or LIM:CLE: AUTO 1.</p>
-103	<p>Invalid separator</p> <p>An invalid separator was found in the command string. You may have used a comma instead of a colon, semicolon, or blank space; or you may have used a blank space instead of a comma.</p> <p>For example, OUTP:ROSC,1.</p>
-105	<p>GET not allowed</p> <p>A Group Execute Trigger (GET) is not allowed within a command string.</p>
-108	<p>Parameter not allowed</p> <p>More parameters were received than expected for the command. You may have entered an extra parameter, or added a parameter to a command that does not accept a parameter.</p> <p>For example, CAL 10.</p>
-109	<p>Missing parameter</p> <p>Fewer parameters were received than expected for the command. You omitted one or more parameters that are required for this command.</p> <p>For example, AVER:COUN.</p>
-112	<p>Program mnemonic too long</p> <p>A command header was received which contained more than the maximum 12 characters allowed.</p> <p>For example, SENSEAVERAGECOUNT 8.</p>
-113	<p>Undefined header</p> <p>A command was received that is not valid for this power sensor. You may have misspelled the command, it may not be a valid command or you may have the wrong interface selected. If you are using the short form of the command, remember that it may contain up to four letters.</p> <p>For example, TRIG:SOUR IMM.</p>

-121	<p>Invalid character in number An invalid character was found in the number specified for a parameter value. For example, SENS:AVER:COUN 128#H.</p>
-123	<p>Exponent too large A numeric parameter was found whose exponent was larger than 32,000. For example, SENS:COUN 1E34000.</p>
-124	<p>Too many digits A numeric parameter was found whose mantissa contained more than 255 digits, excluding leading zeros.</p>
-128	<p>Numeric data not allowed A numeric value was received within a command which does not accept a numeric value. For example, MEM:CLE 24.</p>
-131	<p>Invalid suffix A suffix was incorrectly specified for a numeric parameter. You may have misspelled the suffix. For example, SENS:FREQ 200KZ.</p>
-134	<p>Suffix too long A suffix used contained more than 12 characters. For example, SENS:FREQ 2MHZZZZZZZZZZ.</p>
-138	<p>Suffix not allowed A suffix was received following a numeric parameter which does not accept a suffix. For example, INIT:CONT 0Hz.</p>
-148	<p>Character data not allowed A discrete parameter was received but a character string or a numeric parameter was expected. Check the list of parameters to verify that you have used a valid parameter type. For example, MEM:CLE CUSTOM_1.</p>
-151	<p>Invalid string data An invalid string was received. Check to see if you have enclosed the character string in single or double quotes. For example, MEM:CLE "CUSTOM_1.</p>
-158	<p>String data not allowed A character string was received but is not allowed for the command. Check the list of parameters to verify that you have used a valid parameter type. For example, LIM:STAT 'ON'.</p>
-161	<p>Invalid block data A block data element was expected but was invalid for some reason. For example, *DDT #15FET. The 5 in the string indicates that 5 characters should follow, whereas in this example there are only 3.</p>

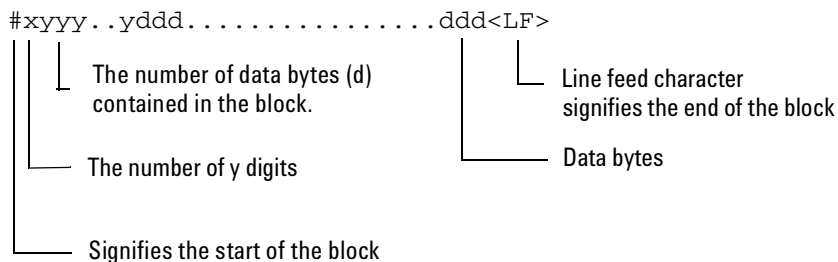
-168	<p>Block data not allowed</p> <p>A legal block data element was encountered but not allowed by the power sensor at this point.</p> <p>For example SYST:LANG #15FETC?.</p>
-178	<p>Expression data not allowed</p> <p>A legal expression data was encountered but not allowed by the power sensor at this point.</p> <p>For example SYST:LANG (5+2).</p>
-211	<p>Trigger ignored</p> <p>Indicates that *TRG, or TRIG:IMM was received and recognized by the device but was ignored because the power sensor was not in the wait for trigger state.</p>
-213	<p>Init ignored</p> <p>Indicates that a request for a measurement initiation was ignored as the power sensor was already initiated.</p> <p>For example, INIT:CONT ON</p> <p>INIT.</p>
-214	<p>Trigger deadlock</p> <p>TRIG:SOUR was set to HOLD or BUS and a READ? or MEASure? was attempted, expecting TRIG:SOUR to be set to IMMEDIATE.</p>
-220	<p>Parameter error;Frequency list must be in ascending order.</p> <p>Indicates that the frequencies entered using the</p> <p>MEMory:TABLE:FREQuency command are not in ascending order.</p>
-221	<p>Settings conflict</p> <p>This message occurs under a variety of conflicting conditions. The following list gives a few examples of where this error may occur:</p> <p>If the READ? parameters do not match the current settings.</p> <p>If you are in fast mode and attempting to switch on for example, averaging, duty cycle or limits.</p> <p>Trying to clear a sensor calibration table when none is selected.</p>
-222	<p>Data out of range</p> <p>A numeric parameter value is outside the valid range for the command.</p> <p>For example, SENS:FREQ 2KHZ.</p>
-224	<p>Illegal parameter value</p> <p>A discrete parameter was received which was not a valid choice for the command. You may have used an invalid parameter choice.</p> <p>For example, TRIG:SOUR EXT.</p>

-226	Lists not same length This occurs when SENSE:CORRection:CSET[2]:STATe is set to ON and the frequency and calibration/offset lists do not correspond in length.
-230	Data corrupt or stale This occurs when a FETC? is attempted and either a reset has been received or the power sensor state has changed such that the current measurement is invalidated (for example, a change of frequency setting or triggering conditions).
-231	Data questionable;CAL ERROR Power sensor calibration failed.
-231	Data questionable;ZERO ERROR Power sensor zeroing failed. The most likely cause is attempting to zero when some power signal is being applied to the power sensor.
-310	System error;Sensor non-volatile memory Read Failed - critical data not found or unreadable This indicates a failure with your U2000 Series USB power sensors. Refer to your power sensor manual for details on returning it for repair.
-321	Out of memory The power sensor required more memory than was available to run an internal operation.
-330	Self-test Failed; The -330, "Self-test Failed" errors indicate that you have a problem with your power sensor. Refer to Contacting Agilent Technologies for details of what to do with your faulty power sensor.
-330	Self-test Failed;Measurement Channel Fault
-330	Self-test Failed;ROM Check Failed
-330	Self-test Failed;RAM Check Failed
-350	Queue overflow The error queue is full and another error has occurred which could not be recorded.
-410	Query INTERRUPTED A command was received which sends data to the output buffer, but the output buffer contained data from a previous command (the previous data is not overwritten). The output buffer is cleared when power has been off, or after *RST (reset) command has been executed.
-420	Query UNTERMINATED The power sensor was addressed to talk (that is, to send data over the interface) but a command has not been received which sends data to the output buffer. For example you may have executed a CONFigure command (which does not generate data) and then attempted to read data from the remote interface.

-430	<p>Query DEADLOCKED</p> <p>A command was received which generates too much data to fit in the output buffer and the input buffer is also full. Command execution continues but data is lost.</p>
-440	<p>Query UNTERMINATED after indefinite response</p> <p>The *IDN? command must be the last query command within a command string.</p>

SYSTem:HELP:HEADers?

This query returns a list of all SCPI commands supported by the instrument. Data is returned in IEEE 488.2 arbitrary block program data format as shown in Figure 10-1 below.



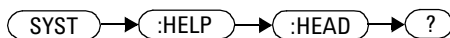
Example: if there are 12435 data bytes, $y = 12435$ and $x = 5$

Figure 10-1 IEEE 488.2 Arbitrary Block Program Data Format

Each point in the trace is represented as an IEEE 754 32 bit floating point number, made up of four bytes in the data block. The MS byte is transmitted first. Each complete block is terminated by a line feed.

Commands are listed in alphabetical order.

Syntax



Example

`SYST:HELP:HEAD?`

This command returns the SCPI commands supported by the instrument.

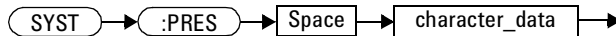
SYSTem:PRESet <character_data>

This command presets the power sensor to values appropriate for measuring the communications format specified by <character_data>. The power sensor is preset to default values if no value or the value DEFault is supplied.

NOTE

DEFault settings apply to both *RST and to SYSTem:PRESet DEFault unless stated otherwise.

Syntax



Parameters

Item	Description	Range of Values
character_data	A communications format which determines the preset values.	DEFault

Example

```
SYST:PRESet DEF
```

This command presets the power sensor with default values. The same default values are set when the parameter is omitted.

Preset Values

DEfault

Table 10-1 shows the power sensor presets when <character_data> is set to DEfault or omitted. Values are shown for all SCPI commands:

Table 10-1 DEfault: Power Sensor Presets

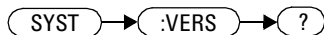
Command	Setting	Comments
CALC[1]:FEED[1]	"POW:AVER"	Select average measurement type
CALC[1]:LIM:CLE:AUTO	ON	Clear limit data at INIT
CALC[1]:LIM:LOW[:DATA]	-90 dBm	Lower limit
CALC[1]:LIM:STAT	OFF	Window limits checking disabled
CALC[1]:LIM:UPP[:DATA]	+90 dBm	
CALC[1]:MATH[:EXPR]	Upper Lower	Math expression
FORM[:READ]:BORD	normal	Binary order
FORM[:READ][:DATA]	ascii	Data format
INIT[1]:CONT	*RST: OFF SYS:PRES ON	Power Sensor in idle state Power Sensor in wait for trigger state
MEM:TABL:SEL	not affected	Active sensor calibration table
[SENSe[1]]:AVER:COUN	4	Filter length
[SENSe[1]]:AVER:COUN:AUTO	ON	Auto-filtering enabled
[SENSe[1]]:AVER:SDET	1	Step detection enabled
[SENSe[1]]:AVER[:STAT]	ON	Averaging enabled
[SENSe[1]]:CORR:CSET[2][:SEL]	not affected	Selected frequency dependent offset table
[SENSe[1]]:CORR:CSET[2]:STAT	not affected	Sensor frequency dependent offset disabled
[SENSe[1]]:CORR:FDOF GAIN4[:INP][:MAGN]	not affected	Return frequency dependent offset
[SENSe[1]]:CORR:GAIN2:STAT	OFF	Channel offset disabled
[SENSe[1]]:CORR:GAIN2:STAT[:INPut][:MAGNitude]	0.0 dB	Enter channel offset value

Command	Setting	Comments
[SENSe[1]]:DET:FUNC	AVER	Measurement mode
[SENSe[1]]:FREQ[:CW]:FIX]	+50.000 MHz	Frequency setting
[SENSe[1]]:MRAT	NORM	Measurement speed
[SENSe[1]]:POW:AC:RANG	upper	Upper range selected
[SENSe[1]]:POW:AC:RANG: AUTO	ON	Auto-ranging selected
TRAC[1]:STAT	OFF	Disable trace capture
TRAC[1]:UNIT	dBm	Trace units
TRIG[1]:DEL:AUTO	ON	Insert settling time delay
TRIG[:SEQ]:SLOP	POS	Trigger event recognized on rising edge
TRIG[:SEQ[1]]:COUN	1	Trigger events for measurement cycle
TRIG[:SEQ[1]]:DEL:AUTO	ON	Enable settling time delay
TRIG[:SEQ[1]]:SOUR	IMM	Trigger source set up
UNIT:POW	dBm	Power units

SYSTEM:VERSion?

This query returns the version of SCPI used in the power sensor. The response is in the form of XXXX.Y, where XXXX is the year and Y is the version number.

Syntax



Example

SYST:VERS?

This command queries which version of SCPI is used in the power sensor.



11 TRIGger Subsystem

TRIGger Subsystem	262
ABORt[1]	263
INITiate Commands	264
INITiate[1]:CONTInuous <boolean>	265
INITiate[1]:IMMEDIATE	267
INITiate:CONTInuous:ALL <boolean>	268
INITiate[1]:CONTInuous:SEQuence[1] <boolean>	270
INITiate[1]:IMMEDIATE:ALL	272
INITiate[1]:IMMEDIATE:SEQuence[1]	273
TRIGger Commands	274
TRIGger[1]:DELay:AUTO <boolean>	275
TRIGger[1]:IMMEDIATE	277
TRIGger[1]:SOURce BUS EXTernal HOLD IMMEDIATE	278
TRIGger[:SEQuence]:SLOPe <character_data>	281
TRIGger[:SEQuence[1]]:COUNT <numeric_value>	283
TRIGger[:SEQuence[1]]:DELay:AUTO <boolean>	285
TRIGger[:SEQuence[1]]:IMMEDIATE	287
TRIGger[:SEQuence[1]]:SOURce BUS EXTernal HOLD IMMEDIATE	288

This chapter explains how the TRIGger command subsystem is used to synchronize device actions with events.



TRIGger Subsystem

The TRIGger subsystem is used to synchronize device actions with events. It includes the ABORT, INITiate and TRIGger commands. These are all at the root level in the command hierarchy but they are grouped here because of their close functional relationship.

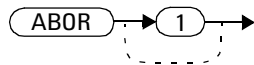
Keyword	Parameter Form	Notes	Page
ABORT [1]		[no query] [non-SCPI]	page 263
INITiate [1]			
:CONTinuous			
:ALL	<boolean>		page 268
:SEQuence [1]	<boolean>		page 270
[:IMMediate]			
:ALL		[no query]	page 272
:SEQuence [1]		[no query]	page 273
TRIGger [1]			
:DELay			
:AUTO	<boolean>		page 275
[:IMMediate]		[no query]	page 277
:SOURce	BUS EXTErnal HOLD IMMediate		page 278
TRIGger			
[:SEQuence]			
:SLOPe	<character_data>		page 281
[:SEQuence [1]]			
:COUNT	<numeric_value>		page 283
:DELay			
:AUTO	<boolean>		page 285
:IMMediate		[no query]	page 287
:SOURce	BUS EXTErnal HOLD IMMediate		page 288

ABORt[1]

This command removes the sensor from the wait for trigger state and places it in the idle state. It does not affect any other settings of the trigger system. When the INITiate command is sent, the trigger system responds as it did before ABORt was executed.

If INITiate:CONTinuous is ON, then after ABORt the current measurement immediately goes into the wait for trigger state.

Syntax



Example

ABOR

This command places the sensor in the idle state.

INITiate Commands

Initiate commands allow you to place the power sensor in the wait for trigger state.

The INITiate commands are overlapped, that is, the power sensor can continue parsing and executing subsequent commands while initiated. Note that the pending operation flag is set, when the power sensor enters an idle state and the flag is cleared when it re-enters the idle state.

The following commands are described in this section:

```
INITiate[1]:CONTInuous <boolean>
```

```
INITiate[1][:IMMEDIATE]
```

```
INITiate:CONTInuous:ALL <boolean>
```

```
INITiate:CONTInuous:SEQuence[1] <boolean>
```

```
INITiate[:IMMEDIATE]:ALL
```

```
INITiate[:IMMEDIATE]:SEQuence[1]
```


INITiate[1]:CONTInuous <boolean>

This command sets the power sensor for either a single trigger cycle or continuous trigger cycles. A trigger cycle means that the power sensor exits the wait for trigger state and starts a measurement.

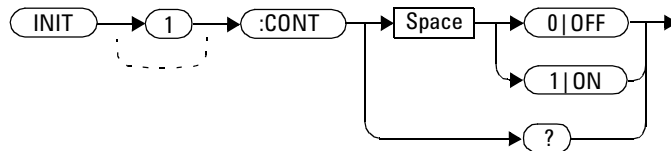
If INITiate:CONTInuous is set to:

- OFF, the trigger system remains in the idle state until it is set to ON, or INITiate:IMMediate is received. Once this trigger cycle is complete the trigger system returns to the idle state.
- ON, the trigger system is initiated and exits the idle state. On completion of each trigger cycle, the trigger system immediately commences another trigger cycle without entering the idle state.

NOTE

This command performs the same function as INITiate:CONTInuous:SEQuence[1] <boolean>.

Syntax



Example

```
INIT1:CONT ON
```

This command places the sensor in the wait for trigger state.

Reset Condition

On reset (*RST), this command is set to OFF.

On preset (SYSTEM:PRESet) and instrument power-up, INITiate:CONTinuous is set to ON.

Query

```
INITiate[1]:CONTinuous?
```

The query enters a 1 or 0 into the output buffer.

- 1 is returned when there is continuous triggering
- 0 is returned when there is only a single trigger

Query Example

```
INIT:CONT?
```

This command queries whether the sensor is set for single or continuous triggering.

INITiate[1][:IMMEDIATE]

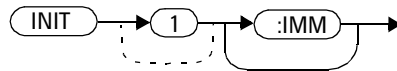
This command sets the power sensor in the wait for trigger state. When a trigger is received, the measurement is taken and the result placed in the power sensor memory. If TRIGger:SOURce is set to IMMEDIATE the measurement begins as soon as INITiate:IMMEDIATE is executed.

Use FETCH? to transfer a measurement from memory to the output buffer. Refer to “FETCH[1]?” on page 74 for further details.

NOTE

This command performs the same function as INITiate:[IMMEDIATE]:SEQUENCE[1].

Syntax



Example

```
INIT1:IMM
```

This command places the sensor in the wait for trigger state.

Error Messages

If the power sensor is not in the idle state or INITiate:CONTInuous is ON, error -213, “INIT ignored” occurs.

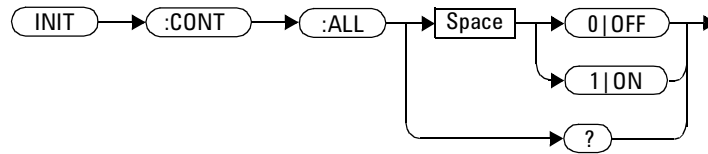
INITiate:CONTInuous:ALL <boolean>

Sets all trigger sequences to be continuously initiated.

If INITiate:CONTInuous:ALL is set to:

- ON, trigger sequences are set to be continuously initiated
- OFF, trigger sequences are not set to be continuously initiated

Syntax



Example

```
INIT:CONT:ALL ON
```

This command sets all trigger sequences to be continuously initiated.

Reset Condition

On reset (*RST), this command is set to OFF.

On preset (SYSTEM:PRESet) and instrument power-up, INITiate:CONTInuous is set to ON.

Query

```
INITiate:CONTInuous:ALL?
```

The query enters a 1 or 0 into the output buffer.

- 1 is returned when trigger sequences are set to be continuous
- 0 is returned when trigger sequences are not set to be continuous

Query Example

```
INIT:CONT:ALL?
```

This command queries whether the sensor is in a wait for trigger state.

INITiate[1]:CONTInuous:SEQUence[1] <boolean>

This command sets the power sensor for either a single trigger cycle or continuous trigger cycles. A trigger cycle means that the power sensor exits the wait for trigger state and starts a measurement.

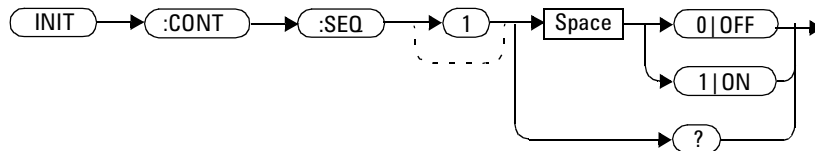
If INITiate:CONTInuous:SEQUence[1] <boolean> is set to:

- OFF, the trigger system remains in the idle state until it is set to ON, or INITiate:IMMediate is received. Once this trigger cycle is complete the trigger system returns to the idle state.
- ON, the trigger system is initiated and exits the idle state. On completion of each trigger cycle, the trigger system immediately commences another trigger cycle without entering the idle state.

NOTE

This command performs the same functions as INITiate[1]:CONTInuous <boolean>.

Syntax



Example

```
INIT:CONT:SEQ1 ON
```

This command places the sensor in a wait for trigger state.

Reset Condition

On reset (*RST), this command is disabled.

On preset (SYSTem:PRESet) and instrument power-up, this command is enabled.

Query

```
INITiate[1]:CONTinuous:SEQuence?
```

The query enters a 1 or 0 into the output buffer.

- 1 is returned when there is continuous triggering
- 0 is returned when there is only a single trigger

Query Example

```
INIT1:CONT:SEQ?
```

This command queries whether the sensor is set for single or continuous triggering.

INITiate[1][:IMMEDIATE]:ALL

This command initiates all trigger sequences.

Syntax



Example

```
INIT:IMM:ALL
```

This command initiates all trigger sequences.

Error Messages

If the power sensor is not in the idle state or INITiate:CONTInuous is ON, error -213, “INIT ignored” occurs.

INITiate[1][:IMMediate]:SEQuence[1]

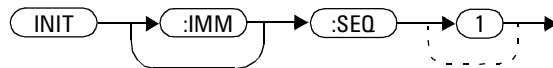
This command sets the power sensor in the wait for trigger state. When a trigger is received, the measurement is taken and the result placed in the power sensor memory. If TRIGger:SOURce is set to IMMEDIATE the measurement begins as soon as INITiate:IMMediate is executed.

Use FETCh? to transfer a measurement from memory to the output buffer. Refer to “FETCh[1]?” on page 74 for further information.

NOTE

This command performs the same function as INITiate[1] : [IMMediate].

Syntax



Example

```
INIT:IMM:SEQ1
```

This command places the sensor in the wait for trigger state.

Error Messages

If the power sensor is not in the “idle” state or INITiate:CONTinuous is ON, error -213, “INIT ignored” occurs.

TRIGger Commands

TRIGger commands control the behavior of the trigger system.

The following commands are described in this section:

```
TRIGger[1]:DElay:AUTO <boolean>
```

```
TRIGger[1]:SOURce BUS|EXTErnal|IMMediate|HOLD
```

```
TRIGger[1][:IMMediate]
```

```
TRIGger:[SEQuence]:SLOPe <character_data>
```

```
TRIGger:SEQuence[1]:COUNT <numeric_value>
```

```
TRIGger:SEQuence[1]:DElay:AUTO <boolean>
```

```
TRIGger:SEQuence[1]:IMMediate
```

```
TRIGger:SEQuence[1]:SOURce  
BUS|EXTErnal|IMMediate|HOLD
```

TRIGger[1]:DELay:AUTO <boolean>

This command is used to determine whether or not there is a settling-time delay before a measurement is made.

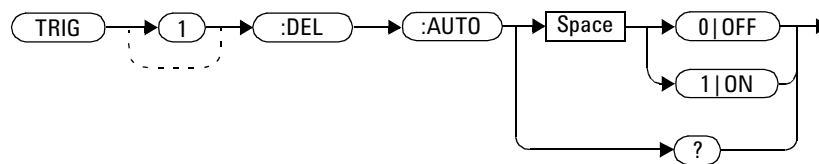
When this command is set to:

- ON, the power sensor inserts a settling-time delay before taking the requested measurement. This settling time allows the internal digital filter to be updated with new values to produce valid, accurate measurement results. The trigger with delay command allows settling time for the internal amplifiers and filters. It does not allow time for power sensor delay.

In cases of large power changes, the delay may not be sufficient for complete settling. Accurate readings can be assured by taking two successive measurements for comparison.

- OFF, the power sensor makes the measurement immediately a trigger is received.

Syntax



Example

```
TRIG:DEL:AUTO ON
```

This command enables a delay on the sensor.

Reset Condition

On reset, TRIGger:DELAy:AUTO is set to ON.

Query

```
TRIGger:DELAy:AUTO?
```

The query enters a 1 or 0 into the output buffer indicating the status of TRIGger:DELAy:AUTO.

- 1 is returned when it is ON
- 0 is returned when it is OFF

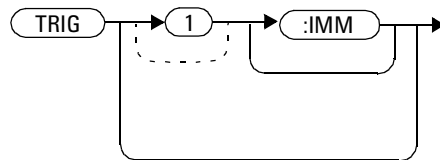
TRIGger[1][:IMMEDIATE]

This command causes a trigger to occur immediately, provided the sensor is in the wait for trigger state. When this command is executed, the measurement result is stored in the power sensor's memory. Use FETCh? to place the measurement result in the output buffer.

NOTE

This command performs the same function as `INITiate[1]:[IMMEDIATE]`.

Syntax



Example

TRIG

This command causes a sensor trigger to occur immediately.

Error Messages

If the power sensor is not in the wait for trigger state, then `TRIGger:IMMEDIATE` causes error -211, "Trigger ignored".

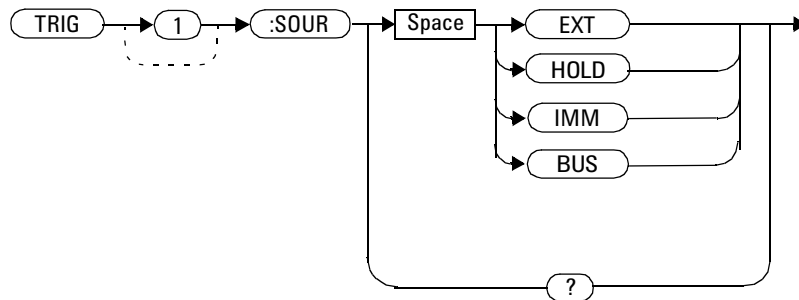
TRIGger[1]:SOURce BUS | EXTernal | HOLD | IMMEDIATE

This command configures the trigger system to respond to the specified source. This command only selects the trigger source. Use the `INITiate` command to place the power sensor in the wait for trigger state.

NOTE

This command has been included for compatibility purposes. It has the same purpose as `TRIGger[:SEquence[1]]:SOURce BUS|EXTernal|HOLD|IMMEDIATE` which should be used in preference.

Syntax



Parameters

Item	Description/Default	Range of Values
source	<p>Available trigger sources:</p> <ul style="list-style-type: none"> BUS: the trigger source is a *TRG common command or the TRIGGER:IMMEDIATE SCPI command. EXTernal: the trigger source is the trigger input in the sensor. HOLD: triggering is suspended. The only way to trigger the power sensor is to use TRIGGER:IMMEDIATE. IMMEDIATE: the trigger system is always true. If INITiate:CONTinuous is ON the power sensor is continually triggering free (free run mode). If an INITiate:IMMEDIATE command is sent a measurement is triggered then the power sensor returns to the idle state. 	EXTernal HOLD IMMEDIATE BUS

NOTE

The trigger source is set to IMMEDIATE on instrument power-up.

The MEASure and CONFigure commands automatically set the trigger source to IMMEDIATE.

The READ? or MEASure commands should not be used if the trigger source is set to HOLD.

Example

```
TRIG:SOUR IMM
```

This command configures the sensor for immediate triggering.

Reset Condition

On reset, the trigger source is set to IMMEDIATE.

Query

TRIGger:SOURce?

The query returns the current trigger source, either IMM, BUS, EXT or HOLD.

Query Example

TRIG:SOUR?

This command queries the sensor's trigger source.

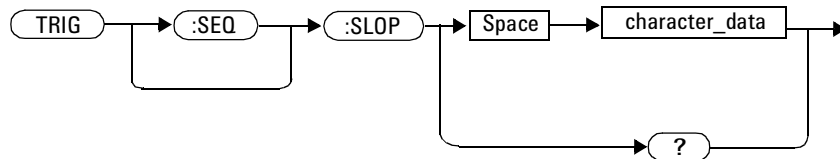
Error Messages

- If the source is changed to EXT and SENS:MRATE has a value of FAST, error -221 “Settings Conflict” occurs.
- If the source is changed to EXT and SENS:DET:FUNC is set to AVERAGE, error -221 “Settings Conflict” occurs.

TRIGger[:SEQuence]:SLOPe <character_data>

This command specifies whether a trigger event is recognized on the rising or falling edge of a signal.

Syntax



Parameters

Item	Description/Default	Range of Values
character_data	How a trigger event is recognized: <ul style="list-style-type: none"> • POSitive: a trigger event is recognized on the rising edge of a signal. • NEGative: a trigger event is recognized on the falling edge of a signal. 	POSitive NEGative

Reset Condition

On reset the value is set to POSitive.

Query

```
TRIGger[:SEquence]:SLOPe?
```

The query returns the current value of <character_data>.

Query Example

```
TRIG:SEQ:SLOP?
```

This command queries the current value of <character_data> for the sensor.

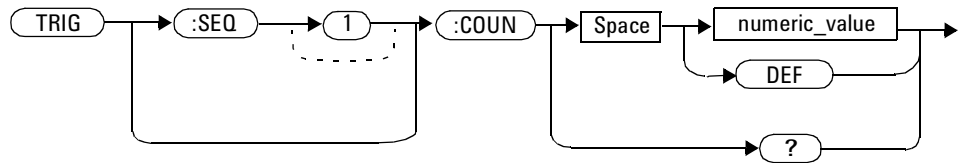
TRIGger[:SEQuence[1]]:COUNt <numeric_value>

This command controls the path of the trigger subsystem in the upward traverse of the wait for trigger state. COUNT loops through the event detection/measurement cycle are performed. That is, COUNT measurements are performed in response to COUNT trigger events.

COUNT can be set to a value >1 only when:

- [SENSE[1]]:MRATE <character_data> is set to FAST
- and
- TRIGger:SOURce set to IMMEDIATE or HOLD.

Syntax



Parameters

Item	Description/Default	Range of Values
numeric_value	The number of triggered events for the measurement cycle. <ul style="list-style-type: none"> • DEF: the default value is 1 	1 to 50 DEF

Example

```
TRIG:SEQ1:COUN 10
```

This command sets the number of triggered events to 10 for the sensor measurement cycle.

Reset Condition

On reset, the value is set to 1.

Query

```
TRIGger:SEQuence[1]:COUNT?
```

The query returns the current setting of trigger events for the sensor.

Query Example

```
TRIG:SEQ1:COUN?
```

This command queries the number of triggered events for the sensor measurement cycle.

Error Messages

If COUNT >1 when [SENSe[1]]:MRATe <character_data> is set to NORMAl or DOUBle, error -221, “Settings Conflict” occurs.

TRIGger[:SEQuence[1]]:DELay:AUTO <boolean>

This command is used to determine whether or not there is a settling-time delay before a measurement is made.

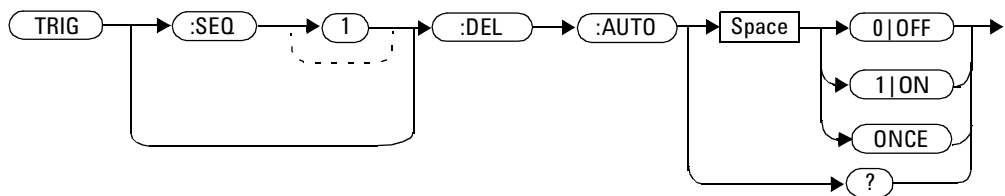
When this command is set to:

- ON, the power sensor inserts a settling-time delay before taking the requested measurement and for subsequent measurements. This settling time allows the internal digital filter to be updated with new values to produce valid, accurate measurement results. The trigger with delay command allows settling time for the internal amplifiers and filters. It does not allow time for power sensor delay.

In cases of large power changes, the delay may not be sufficient for complete settling. Accurate readings can be assured by taking two successive measurements for comparison.

- OFF, no settling-time delay is inserted and the power sensor makes the measurement immediately a trigger is received.
- ONCE, a settling-time delay is inserted before taking the requested measurement, for one measurement only.

Syntax



Example

```
TRIG:SEQ:DEL:AUTO ON
```

This command enables a delay on the sensor.

Reset Condition

On reset, TRIGger:DELAy:AUTO is set to ON.

Query

```
TRIGger:DELAy:AUTO?
```

The query enters a 1 or 0 into the output buffer indicating the status of TRIGger:DELAy:AUTO.

- 1 is returned when it is ON
- 0 is returned when it is OFF

Query Example

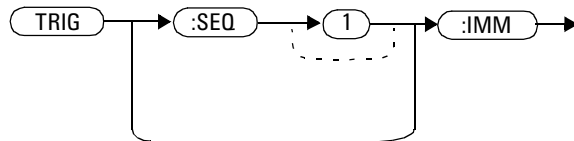
```
TRIG:SEQ1:DEL:AUTO?
```

This command queries the settling-time delay of the sensor.

TRIGger[:SEQuence[1]]:IMMEDIATE

This command provides a one time over-ride of the normal process of the downward path through the wait for trigger state. It causes the immediate exit of the event detection layer if the trigger system is in this layer when the command is received. In other words, the instrument stops waiting for a trigger and takes a measurement ignoring any delay set by TRIG:DElay.

Syntax



Example

TRIG:SEQ:IMM

This command initiates a measurement on the sensor.

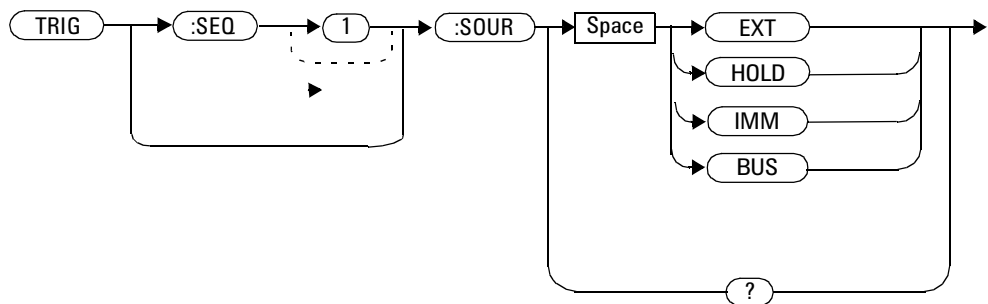
TRIGger[:SEQuence[1]]:SOURce BUS | EXTernal | HOLD | IMMEDIATE

This command configures the trigger system to respond to the specified source. This command only selects the trigger source. Use the `INITiate` command to place the power sensor in the wait for trigger state.

NOTE

This command has the same purpose as `TRIGger[1]:SOURce`
`BUS | EXTernal | HOLD | IMMEDIATE`.

Syntax



Parameters

Item	Description/Default	Range of Values
source	<p>Available trigger sources:</p> <ul style="list-style-type: none"> BUS: the trigger source is a *TRG common command or the TRIGGER:IMMEDIATE SCPI command. EXTernal: the trigger source is the trigger input in the sensor. HOLD: triggering is suspended. The only way to trigger the power sensor is to use TRIGger:IMMEDIATE. IMMEDIATE: the trigger system is always true. If INITiate:CONTINUOUS is ON the power sensor is continually triggering free (free run mode). If an INITiate:IMMEDIATE command is sent a measurement is triggered then the power sensor returns to the idle state. 	EXTERNAL HOLD IMMEDIATE BUS

NOTE

The trigger source is set to IMMEDIATE on instrument power-up.

The MEASURE and CONFIGure commands automatically set the trigger source to IMMEDIATE.

The READ? or MEASURE commands should not be used if the trigger source is set to HOLD.

Example

```
TRIG:SOUR IMM
```

This command configures the sensor for immediate triggering.

Reset Condition

On reset, the trigger source is set to IMMEDIATE.

Query

TRIGger:SEQuence[1]:SOURce?

The query returns the current trigger source.

Query Example

TRIG:SEQ1:SOUR?

This command queries the current trigger source for the sensor.

Error Messages

- If the source is changed to EXT and [SENSe[1]]:MRATe has a value of FAST, error -221 “Settings Conflict” occurs.
- If the source is changed to EXT and [SENSe[1]]:DET:FUNC is set to AVERAge, error -221 “Settings Conflict” occurs.



12 UNIT Subsystem

UNIT Subsystem [292](#)

UNIT[1]:POWer <amplitude_unit> [293](#)

This chapter explains how the `UNIT` command subsystem is used to set the power sensor measurement units to Watts and % (linear), or dBm and dB (logarithmic).



UNIT Subsystem

The UNIT command subsystem:

- Sets power measurement units to dBm or Watts.

This UNIT command has a numeric suffix which determines which window/measurement is set:

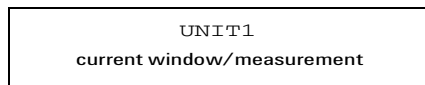


Figure 12-1 Measurement Display UNIT Block Window

The following commands are described in this section:

Keyword	Parameter Form	Notes	Page
UNIT[1] :POWer	<amplitude unit>		page 293

The UNIT:POWer command is as follows:

- If UNIT:POWer is set to dBm.
- If UNIT:POWer is set to W.

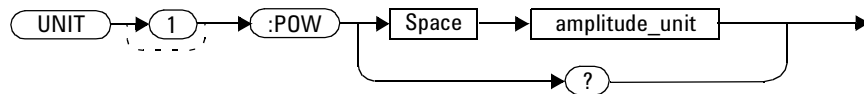
UNIT[1]:POWer <amplitude_unit>

This command sets the power measurement units for a current window/measurement. The power suffix set by UNIT:POWer is used for any command which accepts a numeric value in more than one unit

For single measurement:

- UNIT1:POWer sets the power measurement units for the current window/measurement.

Syntax



Parameters

Item	Description/Default	Range of Values
amplitude_unit	The measurement unit. <ul style="list-style-type: none"> The default unit is dBm 	W DBM

Example

```
UNIT1:POW DBM
```

This command sets the power measurement units for the current window/measurement.

Reset Condition

On reset, all windows/measurements are set to DBM.

Query

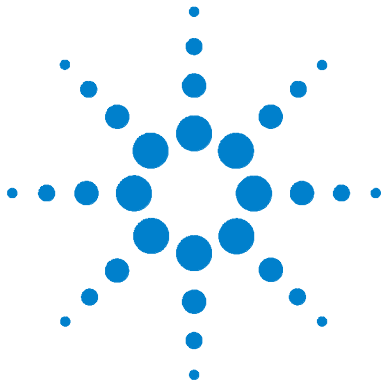
```
UNIT[1]:POWer?
```

The query returns the current setting of the power measurement units.

Query Example

```
UNIT1:POW?
```

This command queries which measurement units are being used on the current window/measurement.



13 IEEE 488.2 Command Reference

SCPI Compliance Information 296

*CLS 297

*ESE <NRf> 298

*ESR? 300

*IDN? 301

*OPC 302

*OPT? 303

*RCL <NRf> 304

*RST 305

*SAV <NRf> 306

*SRE <NRf> 307

*STB? 309

*TRG 311

*TST? 312

*WAI 313

USBTMC/USB488 Universal Commands 314

This chapter contains information about the IEEE 488.2 Common Commands that the power sensor supports.



SCPI Compliance Information

This chapter contains information about the SCPI Common (*) Commands that the power sensor supports. It also describes the USBTMC/USB488 Universal Command statements which form the nucleus of USB programming; they are understood by all instruments in the network. When combined with programming language codes, they provide all management and data communication instructions for the system.

The IEEE-488.2 Common Command descriptions are listed below in alphabetical order.

*CLS	Clear Status	page 297
*ESE and *ESE?	Event Status Enable	page 298
*ESR?	Event Status Register	page 300
*IDN?	Identify	page 301
*OPC and *OPC?	Operation Complete	page 302
*OPT?	Options	page 303
*RCL	Recall	page 304
*RST	Reset	page 305
*SAV	Save	page 306
*SRE and *SRE?	Service Request Enable	page 307
*STB?	Status Byte	page 309
*TRG	Trigger	page 311
*TST?	Test	page 312
*WAI	Wait	page 313

*CLS

The *CLS (Clear Status) command clears the status data structures. The SCPI registers (Questionable Status, Operation Status and all the other SCPI registers), the Standard Event Status Register, the Status Byte, and the Error/Event Queue are all cleared.

Syntax

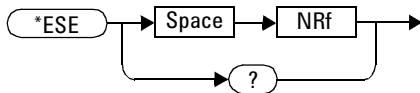
*CLS →

***ESE <Nrf>**

The *ESE (Event Status Enable) <Nrf> command sets the Standard Event Status Enable Register. This register contains a mask value for the bits to be enabled in the Standard Event Status Register. A 1 in the Enable Register enables the corresponding bit in the Status Register, a 0 disables the bit. The parameter value, when rounded to an integer and expressed in base 2, represents the bit values of the Standard Event Status Enable Register. [Table 13-1](#) shows the contents of this register.

Table 13-1 *ESE Mapping

Bit	Weight	Meaning
0	1	Operation Complete
1	2	Request Control (not used)
2	4	Query Error
3	8	Device Dependent Error
4	16	Execution Error
5	32	Command Error
6	64	Not used
7	128	Power On

Syntax

Parameters

Type	Description/Default	Range of Values
NRf	A value used to set the Standard Event Status Enable Register.	0 - 255

Query

*ESE?

The query returns the current contents of the Standard Event Status Enable Register. The format of the return is <NR1> in the range of 0 to 255.

***ESR?**

The *ESR? query returns the contents of the Standard Event Status Register then clears it. The format of the return is <NR1> in the range of 0 to 255. [Table 13-2](#) shows the contents of this register.

Table 13-2 *ESR? Mapping

Bit	Weight	Meaning
0	1	Operation Complete
1	2	Not used
2	4	Query Error
3	8	Device Dependent Error
4	16	Execution Error
5	32	Command Error
6	64	Not used
7	128	Power On

Syntax

*IDN?

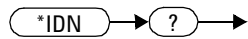
The *IDN? query allows the power sensor to identify itself. The string returned is:

```
Agilent Technologies,U200XA,<serial number>,A.XX.YY
```

where:

- <serial number> uniquely identifies each power sensor.
- represents the firmware revision with XX and YY representing the major and minor revisions respectively.

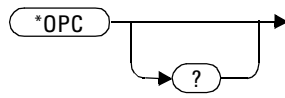
Syntax



*OPC

The *OPC (Operation Complete) command causes the power sensor to set the operation complete bit in the Standard Event Status Register when all pending device operations have completed.

Syntax



Query

*OPC?

The query places an ASCII 1 in the output queue when all pending device operations have completed.

*OPT?

The *OPT? query reports the options installed in the power sensor and returns:

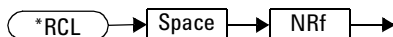
- " " empty string for a standard instrument.

Syntax



***RCL <NRf>**

The *RCL <NRf> (ReCaLI) command restores the state of the power sensor from the specified save/recall register. An instrument setup must have been stored previously in the specified register.

Syntax**Parameters**

Type	Description/Default	Range of Values
NRf	The number of the register to be recalled.	1 - 10

Error Message

- If the register does not contain a saved state, error -224, “Illegal parameter value” occurs.

*RST

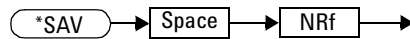
The *RST (ReSeT) command places the power sensor in a known state. Refer to “[SYSTem:PRESet <character_data>](#)” on page 257 for information on reset values.

Syntax

*RST →

***SAV <NRf>**

The *SAV <NRf> (SAVe) command stores the current state of the power sensor in the specified register.

Syntax**Parameters**

Item	Description/Default	Range of Values
NRf	The number of the register that the current state of the power sensor is to be saved to.	1 - 10

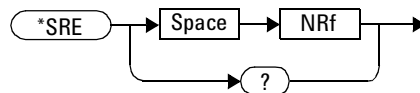
*SRE <NRf>

The *SRE <NRf> command sets the Service Request Enable register bits. This register contains a mask value for the bits to be enabled in the Status Byte Register. A 1 in the Enable Register enables the corresponding bit in the Status Byte Register; a 0 disables the bit. The parameter value, when rounded to an integer and expressed in base 2, represents the bits 0 to 5 and bit 7 of the Service Request Enable Register. Bit 6 is always 0. [Table 13-3](#) shows the contents of this register. Refer to the pullout at the end of Chapter 10 for further information.

Table 13-3 *SRE Mapping

Bit	Weight	Meaning
0	1	Not used
1	2	Not used
2	4	Device Dependent
3	8	QUEStionable Status Summary
4	16	Message Available
5	32	Event Status Bit
6	64	Not used
7	128	OPERation Status Summary

Syntax



Parameters

Type	Description/Default	Range of Values
NRf	A value used to set the Service Request Enable Register.	0 to 255

Query

*SRE?

The query returns the contents of bits 0 to 5 and bit 7 of the Service Request Enable Register. The format of the return is <NR1> in the ranges of 0 to 63 or 128 to 191 (that is, bit 6 is always 0).

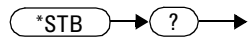
***STB?**

The *STB? (STatus Byte) query returns bit 0 to 5 and bit 7 of the power sensor's status byte and returns the Master Summary Status (MSS) as bit 6. The MSS is the inclusive OR of the bitwise combination (excluding bit 6) of the Status Byte and the Service Request Enable registers. The format of the return is <NR1> in the ranges of 0 to 255. [Table 13-4](#) shows the contents of this register. Refer to the Status Block Diagram at the end of [Chapter 9](#), "STATus Subsystem" for further information.

Table 13-4 *STB? Mapping

Bit	Weight	Meaning
0	1	Not used
1	2	Device Dependent 0 - No device status conditions have occurred 1 - A device status condition has occurred
2	4	Error/Event Queue 0 - Queue empty 1 - Queue not empty
3	8	Questionable Status Summary 0 - No QUEStionable status conditions have occurred 1 - A QUEStionable status condition has occurred
4	16	Message Available 0 - no output messages are ready 1 - an output message is ready
5	32	Event Status Bit 0 - no event status conditions have occurred 1 - an event status condition has occurred
6	64	Master Summary Status 0 - power sensor not requesting service 1 - there is at least one reason for requesting service
7	128	Operation Status Summary 0 - No OPERation status conditions have occurred 1 - An OPERation status condition has occurred

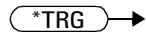
Syntax



*TRG

The *TRG (TRiGger) command triggers all channels that are in the wait for trigger state.

Syntax

 *TRG →

Error Message

- If TRIGger:SOURce is not set to BUS, error -211, “Trigger ignored” occurs.
- If the power sensor is not in the wait-for-trigger state, error -211, “Trigger ignored” occurs.

*TST?

The *TST? (TeST) query causes the power sensor to perform the self test. The test takes approximately 40 seconds.

The result of the test is placed in the output queue.

- 0 is returned if the test passes
- 1 if the test fails

Syntax



*WAI

The *WAI (WAIt) command causes the power sensor to wait until either:

- All pending operations are complete
- The device clear command is received
- Power is cycled

before executing any subsequent commands or queries.

Syntax

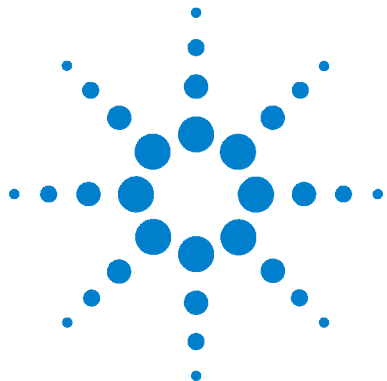
*WAI →

USBTMC/USB488 Universal Commands

DCL

The DCL (Device Clear) command causes all USB instruments to assume a cleared condition. The definition of device clear is unique for each instrument. For the power sensor:

- All pending operations are halted, that is, *OPC? and *WAI.
- The parser (the software that interprets the programming codes) is reset and now expects to receive the first character of a programming code.
- The output buffer is cleared.



A

Calibration Factor Block Layout

Calibration Factor Block Layout [A-2](#)

This chapter contains information on the calibration factor block layout for U2000 Series USB power sensors.



Calibration Factor Block Layout

The following tables provide information on the calibration factor block layout for U2000 Series USB power sensors. The information relates to service commands is described in [Chapter 8](#), “SERVice Subsystem”.

Table A-5 Calibration Factor Block Layout: U2000 Series USB power sensors

U2000 Series USB Power Sensors: Calibration Factor Block Layout	No. Bytes	Contents	Data Format	Data Range	Units	Notes
Header:						
Number of frequency points	2	-	16 bit integer		None	
Number of power low points	2	-	16 bit integer		None	
Number of power high points	2	-	16 bit integer		None	
For Each Table (tables are in the order of lower to upper):						
Power, low	4	-	floating-point	-127.9 to +127.9	dBm	Power for low power flatness.
Power, high	4	-	floating-point	-127.9 to +127.9	dBm	Power for high power flatness.
Frequency (point '0')	4	-	floating-point	0 to $10^{38.53}$	None	Freq in Hz
Cal factor (low power)	4	-	floating-point	0.25 to 3	None	Power (in watts) is divided by this value.
Cal factor (high power)	4	-	floating-point	0.25 to 3	None	Power (in watts) is divided by this value.
....						
Frequency (point 'N')	4	-	floating-point	0 to $10^{38.53}$	None	Freq in Hz

U2000 Series USB Power Sensors: Calibration Factor Block Layout	No. Bytes	Contents	Data Format	Data Range	Units	Notes
Cal factor (low power)	4	-	floating-point	0.25 to 3	None	Power (in watts) is divided by this value.
Cal factor (high power)	4	-	floating-point	0.25 to 3	None	Power (in watts) is divided by this value.

A Calibration Factor Block Layout

www.agilent.com

Contact us

To obtain service, warranty or technical support assistance, contact us at the following phone numbers:

United States:

(tel) 800 829 4444 (fax) 800 829 4433

Canada:

(tel) 877 894 4414 (fax) 800 746 4866

China:

(tel) 800 810 0189 (fax) 800 820 2816

Europe:

(tel) 31 20 547 2111

Japan:

(tel) (81) 426 56 7832 (fax) (81) 426 56 7840

Korea:

(tel) (080) 769 0800 (fax) (080) 769 0900

Latin America:

(tel) (305) 269 7500

Taiwan:

(tel) 0800 047 866 (fax) 0800 286 331

Other Asia Pacific Countries:

(tel) (65) 6375 8100 (fax) (65) 6755 0042

Or visit Agilent worldwide Web at:

www.agilent.com/find/assist

Product specifications and descriptions in this document are subject to change without notice.

© Agilent Technologies, Inc. 2007

Printed in Malaysia

First Edition, July 9, 2007

U2000-90411



Agilent Technologies

www.agilent.com

© Agilent Technologies, Inc. 2007

Printed in Malaysia
First Edition, July 9, 2007

U2000-90411

