S8723X Series USB CW Power Sensor

Programming Manual

Saluki Technology Inc.

The document applies to the continuous wave USB power sensors of the following models:

- S87230 continuous wave USB power sensor (9kHz - 6GHz).
- S87231 continuous wave USB power sensor (10MHz - 18GHz).
- S87232 continuous wave USB power sensor (50MHz - 26.5GHz).
- S87233 continuous wave USB power sensor (50MHz - 40GHz).

Preface

Thanks for choosing S8723X series USB CW power sensor produced by Saluki Technology Inc. Please read this manual carefully for your convenience.

We devote ourselves to meeting your demands, providing you high-quality measuring instrument and the best after-sales service. We persist with “superior quality and considerate service”, and are committed to offering satisfactory products and service for our clients.

Manual No.

S8723X-03-02

Version

Rev01 2019.12

Saluki Technology

Manual Authorization

Product Quality Certificate

The product meets the indicator requirements of the manual at the time of delivery. Calibration and measurement are completed by the measuring organization with qualifications specified by the state, and relevant data are provided for reference.

Quality/Environment Management

Research, development, manufacturing and testing of the product comply with the requirements of the quality and environmental management system.

Contacts

Service Tel:	886.909 602 109
Website:	www.salukitec.com
Email:	sales@salukitec.com
Address:	No. 367 Fuxing N Road, Taipei 105, Taiwan (R.O.C.)

CONTENTS

Chapter 1 About This Manual..... 6
1.1 About This Manual..... 6
1.2 Related Documents..... 7
Chapter 2 Remote Control..... 8
2.1 Remote Control Basis..... 8
2.1.1 Remote Interface..... 8
2.1.2 Message..... 11
2.1.3 Power on/off..... 12
2.1.4 Command Sequence and Synchronization..... 21
2.1.5 Status Reporting System..... 23
2.1.6 Programming Considerations..... 31
2.2 Remote Interface and Its Configuration..... 31
2.3 I/O Library..... 35
2.3.1 Overview of I/O Library..... 35
2.3.2 Installation and Configuration of I/O Library..... 36
2.4 Zeroing..... 36
2.5 Measurement..... 37
2.6 Use of FDO Table..... 37
2.6.1 Overview..... 37
2.6.2 Entering in FDO Table..... 37
2.6.3 Selecting FDO Table..... 38
2.6.4 Enabling FDO Table..... 38
2.6.5 Measurement Application..... 38
2.6.6 Specific Application..... 38
2.7 Setting of Display Resolution..... 39
2.8 Setting of Average..... 39
2.9 Setting of Range..... 39
2.10 Setting of Offset..... 39
2.11 Setting of Measurement Limit..... 40
2.12 Status Reporting..... 40
2.13 Save & Recall..... 41
Chapter 3 SCPI..... 42
3.1 Command Description..... 42

3.2 Common Command (IEEE488.2 Command)..... 42
3.3 Instrument Subsystem Command..... 47
3.3.1 Calculation(CALCulate)..... 47
3.3.2 Calibration (CALibration)..... 52
3.3.3 Measurement (CONFigure/FETCh/READ/MEASure)..... 53
3.3.4 Format (FORMat)..... 56
3.3.5 Memory (MEMory/MMEMory)..... 57
3.3.6 Sense (SENSe)..... 62
3.3.7 Status (STATus)..... 66
3.3.8 System (SYSTem)..... 95
3.3.9 Trigger (INITiate/TRIGger)..... 96
3.3.10 Unit (UNIT)..... 98
3.3.11 Service (SERVice)..... 98
Chapter 4 Programming Example..... 101
4.1 Basic Operation Example..... 101
4.1.1 VISA Library..... 101
4.1.2 Example Running Environment..... 102
4.1.3 Initialization and Default Status Setting..... 102
4.1.4 Sending of Setting Command..... 103
4.1.5 Reading of Instrument Status..... 103
4.1.6 Synchronization of command..... 104
4.2 Advanced Operation Example..... 105
4.2.1 USBTMC Remote Control Example..... 105
Chapter 5 Error Description..... 109
5.1 Error Information..... 109
5.1.1 Local Error Information..... 109
5.1.2 Remote Control Error Information..... 109
5.2 Repair Method..... 113
5.2.1 Contact Us..... 113
5.2.2 Packing and Delivery..... 113
Appendixes..... 115
Appendix A Lookup Table of the SCPI by Subsystem..... 115

Chapter 1 About This Manual

This chapter introduces the function, compositions and main content of the Programming Manual of the S87230 series USB CW power sensor (hereinafter rerferred to as S87230) as well as other related documents provided to the user.

1.1 About This Manual

This manual introduces the remote control and the SCPI operation method of the S87230 series CW power sensor, as well as the programming examples and the basic concept of the I/O function library to facilitate the user to quickly master the programming method. To facilitate your familiarity with the instrument, please read this manual carefully before operating the instrument, and then follow the instructions of manual.

SCPI (Standard Commands for Programmable Instruments) defines standards and methods for remote control of the instruments, and it is also the programming language for programmable instruments for electronic test and measurement. The SCPI is based on the specifications and types in IEEE-488.2. For details, please visit http://www.scpiconsortium.org.

This manual describes in detail the SCPIs of the S87230 series CW power sensor. The chapters of the Programming Manual include:

- **Remote Control**

This chapter introduces the remote control methods of the instrument so that the user can rapidly master the method to control the instrument in a remote way. It is further divided into the following three sections: remote control basis, which introduces the concepts related to remote control, software configuration, remote interface, SCPI, etc.; instrument interface configuration method, which introduces the connection method and software configuration method of the remote interface of the S87230 series USB CW power sensor; the I/O function library, which introduces the basic concept of the instrument driver and the basic installation and configuration of the IVI-COM/IVI-C driver.

- **SCPI**

The common command, instrument-specific command and compatibility command are introduced by category, and functions, parameters, and examples of the SCPI are described one by one.

- **Programming Examples**

The basic programming examples and advanced programming examples are given and described in the form of explanatory note and example code, so as to facilitate the user to quickly master the programming method of the instrument.

- **Error Description**

This chapter includes error information description and repair methods.

- **Appendixes**

This chapter provides the necessary remote control reference information of the S87230 series USB CW power sensor, including the SCPI lookup table.

1.2 Related Documents

The product document related to S87230 series USB CW power sensor includes:

- Quick Start Guide
- User Manual
- Programming Manual

Quick Start Guide

This manual introduces the settings of the instrument as well as the basic operating methods of measurement with the aim of enabling users to quickly understand the features and basic local and remote control operation of the instrument. Main chapters included in this manual are as follows::

- About This Manual
- Preparation before Use
- Typical Applications
- Getting Help

User Manual

This manual gives a detailed introduction of features and operation methods of the instrument, including information about configuration, measurement, remote control, maintenance, etc. so as to provide users with an all-round understanding of the features of the instrument and aid users in learning the most common test procedures. Main chapters included in this manual are as follows:

- About This Manual
- Overview
- Start Guide
- Operation Guide
- Remote Control
- Fault Diagnosis and Repair
- Technical specifications

Programming Manual

This manual describes the basics of remote control programming, basics of SCPI, SCPIs, examples of programming, and I/O driver library, for the purpose of guiding the user to master the SCPIs and methods of the instrument quickly and comprehensively. Main chapters included in this manual are as follows:

- About This Manual
- Remote Control
- SCPI
- Programming Examples
- Error Description
- Appendixes

Chapter 2 Remote Control

This chapter introduces the remote control basis as well as the remote interface and its configuration method of the S8723X series USB CW power sensor, and also briefly describes the concept and classification of the I/O driver library, so that the user can have a preliminary knowledge about the remote control of this instrument. The specific content includes:

- Remote Control Basis
- Remote Interface and its Configuration
- I/O library
- Zeroing
- Measurement
- Use of FDO table
- Setting of Display Resolution
- Setting of Average
- Setting of Range
- Setting of offset
- Setting of measurement limit
- Status reporting
- Save recall

2.1 Remote Control Basis

2.1.1 Remote Interface

The instrument with remote control functions generally supports two kinds of remote interfaces: LAN and GPIB.

The type of port supported by the instrument will be determined by its own functions.

The description of the remote interface and associated VISA addressing string is as shown in the following table:

Table 2.1 Type of the Remote Interface and VISA Addressing String

Remote Interface	VISA Addressing String	Description
LAN (Local Area Network)	**VXI-11 protocol:** TCPIP::host_address[::LAN_device_name][::INSTR] **Raw socket protocol:** TCPIP::host_address::port::SOCKET	Controller realizes remote control by connecting the instrument via the network port on the rear panel of the instrument. For the specific protocol, please refer to: 2.1.1.1 LAN interface
GPIB (IEC/IEEE Bus Interface)	GPIB::primary address[::INSTR]	Controller realizes remote control by connecting the instrument via the port on the rear panel of the instrument. Compliance with the IEC 625.1/IEEE 418 bus interface standard. For details, please refer to: 2.1.1.2 GPIB interface

RS-232 (RecommendedStandard-232)		Instrument's rear panel port For details, please refer to: 2.1.1.3 RS-232 interface
USB (Universal Serial Bus)	USB::<vendor ID>::<product_ID>::<serial_number> [::INSTR]	Instrument's rear panel port For details, please refer to: 2.1.1.4 USB interface

2.1.1.1 LAN Interface

The instrument is available for remote control through the 10Base-T and 100Base-T LAN computers. The instruments can be combined into a system within the LAN, and uniformly controlled by the LAN computers. In order to realize the remote control within the LAN, the instrument shall be preinstalled with the port connector, network card and relevant network protocol, and configured with relevant network service. And, the controller computer within the LAN shall also be preinstalled with the instrument control software and VISA library. The three working modes of the network card include:

- 10 Mbit/s Ethernet IEEE802.3;
- 100 Mbit/s Ethernet IEEE802.3u;
- 1 Gbit/s Ethernet IEEE802.3ab.

The controller computer and the instrument shall be connected with a common TCP/IP protocol network through the network port. The cable between the computer and the instrument is a commercial RJ45 cable (shielded or unshielded CAT 5 twisted pair). During data transmission, data packet transmission will be adopted, and LAN transmission is faster. Generally, the cable between the computer and the instrument shall not be longer than 100 m (100Base-T and 10Base-T). For more information about the LAN communication, please refer to: http://www.ieee.org.

The knowledge of LAN interface is introduced hereinafter.

1) IP address

The physical connection of the network shall be smooth when the instrument is subject to remote control by LAN. The address of the instrument can be set to the subnet where the main control computer is located via menu "Local IP". For example, if the IP address of the main control computer is 192.168.12.0, the IP address of the instrument shall be set to 192.168.12.XXX, and XXX is between 1 and 255.

Only the IP address is required to establish a network connection. The VISA addressing string is as follows:

TCPIP::host address[::LAN device name][::INSTR] or

TCPIP::host address::port::SOCKET

Where,

- TCPIP - network protocol used;
- host address - IP address or host name of the instrument, for identification and control of the controlled instrument;
- The LAN device name defines the handle number of the protocol and sub-device (optional);

----The VXI-11 protocol is adopted for the 0# equipment;

----The newer high speed LAN instrument protocol is adopted for the 0# high speed LAN instrument;

- The INSTR represents the instrument resource type (optional);

- The port represents the socket port number;
- SOCKET - raw socket resource class.

Example:

- The IP address of the instrument is 192.1.2.3, and the valid resource string of the VXI-11 protocol is:

TCPIP::192.1.2.3::INSTR

- When the raw socket connection is created, the following addressing string can be used:

TCPIP::192.1.2.3::5000::SOCKET

Note

Method for identification of multiple instruments in the remote control system

If multiple instruments are connected to the network, they can be identified by their individual IP address and associated resource string. The main control computer uses the respective VISA resource string for instrument identification.

2) VXI-11 protocol

The VXI-11 standard is based on the ONC RPC (Open Network Computing Remote Procedure Call) protocol, which is the network/transport layer of the TCP/IP protocol. The TCP/IP network protocol and relevant network service have been configured in advance. During communication, this connection-oriented communication can follow a sequential exchange and identify the interruption of the connection, thus ensuring no information loss.

3) Socket communication

The TCP/IP protocol connects the signal sources in the network through the LAN socket. As a basic computer network programming method, the socket enables applications with different hardware and operating systems to communicate in the network, This method enables two-way communication between the instrument and the computer via port.

The socket is a special software class that defines the necessary information for network communication such as IP address and device port number and integrates some basic network programming operations. Sockets can be used in the operating system installed with a packaged library. UNIX Berkeley socket and Winsock are commonly used.

The socket in the instrument is compatible with Berkeley socket and Winsock through Application Program Interface (API). In addition, other standard sockets are also compatible through the API. When the instrument is controlled by the SCPI, the socket program created in the program will issue the command. Before using the LAN socket, the socket port number of the instrument shall be set in advance. For this instrument, the socket port number is 5000.

2.1.1.2 GPIB Interface

The GPIB interface is a widely-used instrument remote interface currently, which can be connected with different kinds of instruments through the GPIB cable and can establish the test system with the main control computer. To realize remote control, the main control computer shall be preinstalled with the GPIB bus card, driver and VISA library. During communication, the main control computer will address the controlled instrument through the GPIB bus address firstly. The user can set the GPIB address and ID for querying strings, and the GPIB communication

language can be set to the SCPI form by default.

The operation of the GPIB and its relevant interface is defined and described in details in the ANSI/IEEE standard 488.1-2003 and the ANSI/IEEE standard 488.2-1992. For details of the standard, please refer to the IEEE website: http://www.ieee.org.

As the GPIB processes information in bytes and the data transmission rate can reach 8 MBps, the GPIB data transmission is faster. As the data transmission speed is restricted by the distance between the equipment/system and the computer, attention shall be paid to the followings during the GPIB connection:

- Up to 15 instruments may be set up through the GPIB interface;
- The total length of the transmission cable shall not exceed 15 m, or shall not exceed twice of number of instruments in the system. Generally, the maximum length of the transmission cable between the equipment shall not exceed 2 m;
- If multiple instruments are connected in parallel, the "Or" connecting cables should be used;
- The terminal of the IEC bus cable shall be connected with the instrument or the controller computer.

2.1.1.3 RS232 Interface

The RS-232 is a traditional remote control method. As only one bit of data is sent and received at a time, the transmission rate is slower than that of the GPIB or the LAN, which is rarely used currently. Similar to the GPIB and the LAN, the instrument parameters, such as baud rate, shall be set when establishing communication, so as to match the parameters of the main control computer. The RS-232 transmits the SCPI character in the form of the ASCII code.

2.1.1.4 USB Interface

To realize the USB remote control, it is necessary to connect the computer with the instrument via a USB B-type interface in advance, and preinstall the VISA library. After that, the VISA will automatically test and configure the instrument to establish the USB connection, without the necessity of entry of the instrument address string or installation of an individual driver.

USB address:

Format of the addressing string: USB::<vendor ID>::<product ID>::<serial number>[::INSTR]

Where,

- <vendor ID> ID of vendor;
- <product ID> ID of instrument;
- <serial number> Serial number of instrument.

Example:

USB::0x04b4::0x1004::100001::INSTR

0x04b4: ID of vendor; 0x1004: ID of instrument;

100001: Serial number of instrument.

2.1.2 Message

Messages transmitted by data cable fall into the following two categories:

1) Interface Message

During communication between the instrument and the main control computer, it is necessary to pull down the attention line and then the interface message can be transmitted to the instrument through the data line. Only the instrument with the GPIB bus functions can send the interface message.

2) Instrument message

For the structure and syntax of instrument messages, see "2.1.3 SCPIs" for details. Instrument message can be divided into command and instrument response according to the different transmission directions. All remote control interfaces use instrument message in the same method unless otherwise stated.

a. Commands:

A command (programming message) is a message transmitted from the main control computer to the instrument for remote control of instrument functions and query of status information. It falls into the following two categories:

- Based on the impact on the instrument:

Setting command: Change the instrument setting status, e.g. reset the instrument or set the frequency.

Query command: Query and return the data, e.g. identify the instrument or query the parameter values. The query command is always ended with a question mark.

- Based on the definition in the standard:

Common commands: Functions and syntax defined by IEEE488.2 for all types of instruments (if implemented)

for realizing: Standard status register management, reset, self-test etc.

Instrument control command: Instrument-specific command, for realization of instrument functions. For example: set the frequency.

The syntax also follows SCPI specification.

b. Instrument response:

The instrument response (response message and service request) is the query result information sent by the instrument to the computer. This information includes measurement result and instrument status.

2.1.3 Power on/off

2.1.3.1 Brief Introduction to SCPI

SCPI (Standard Commands for Programmable Instruments) is a set of commands established for all instruments based on IEEE488.2 mainly to achieve the universality of SCPI, i.e.the same SCPI is generated and issued for the same function.

The SCPI consists of a command header and one or more parameters which are separated by a space. The command header contains one or more key fields. The command with question mark as postfix is a query command. Commands are divided into common commands and instrument-specific commands that are different in syntactic structure. SCPI has the following features:

1) The SCPI is established for the test functions rather than instrument operation description.

2) The SCPI reduces the repetition of the realization process of similar test functions, thus ensuring the programming compatibility;

3) The remote control message is defined in a sub-layer irrelated with hardware of the communication physical layer;

4) The SCPI is irrelated with the programming methods and languages, and the SCPI test program is easy to be transplanted;

5) The SCPI has scalability, and can adapt to control of different scales of measurement;

6) Scalability makes SCPI a “Live” standard.

If you are interested in learning more about SCPI, please refer to:

IEEE Standard 488.1-2003, IEEE Standard Digital Interface for Programmable Instrumentation. New York, NY, 1998

IEEE Standard 488.2-1992, IEEE Standard Codes, Formats, Protocols and Comment Commands for Use with ANSI/IEEE Std488.1-2003. New York, NY, 1998

Standard Commands for Programmable Instruments(SCPI) VERSION 1999.0.

For details of the SCPI set, classification and description of S87230 series USB power sensor, please refer to:

- “3 SCPI” of this manual;
- “Appendix A Lookup Table of the SCPI by Subsystem” of this manual.

2.1.3.2 SCPI Description

1) General terms

For the purpose of this section, the following terms should apply. It is necessary to know about the exact definitions of these terms for a better understanding of the content in various chapters.

Controller

The controller is any computer used to communicate with the SCPI equipment. The controller may be a personal computer, a small computer or a card inserted onto a cage. Some artificial intelligence equipment can also be used as a controller.

Equipment

The equipment is any device that supports SCPI. Most equipment is electronic measuring or excitation equipment and use the GPIB interface for communication.

Remote control message

The remote control message is a combination of one or more correctly formatted SCPIs. It guides the equipment to measure and output the signal.

Response message

The response message is a data set that specifies the SCPI format. It is always sent from the equipment to the controller or listener to remind the controller of the internal condition or measured value of the equipment.

Command

A command is an instruction in compliance with the SCPI standard. The combination of controller commands forms a message. In general, a command includes the keyword, parameter and punctuation.

Event command

An event-type SCPI can't be queried. An event command generally has no corresponding key settings on front panel. Its function is to trigger an event at a particular moment.

Query

Query is a special command. When the controller is queried, it is necessary to return to the response message in

conformity with syntax requirement of the controller. The query statement is always ended with a question mark.

2) Command type

There are two types of SCPI commands: common commands and instrument-specific commands. Figure 2.1 shows the difference between two commands. Common commands are defined in IEEE488.2 to manage macros, status registers, synchronization, and data storage. As the common command begins with a *, it can be easily distinguished. For example *IDN? , *OPC and *RST are common commands. Common commands don't belong to any instrument-specific command. The instrument uses the same method to interpret them without consideration to the current path setting.

It is very easy to identify instrument-specific commands because they contain a colon (:). The colon is used between the beginning of a command expression and a keyword, for example: FREQuency[:CW?]. Instrument-specific commands are divided into command subsets of corresponding subsystem according to the functional block inside the instrument. For example, the power subsystem (:POWer) contains the power-related command while the status subsystem (:STATus) contains the command for the status control register.

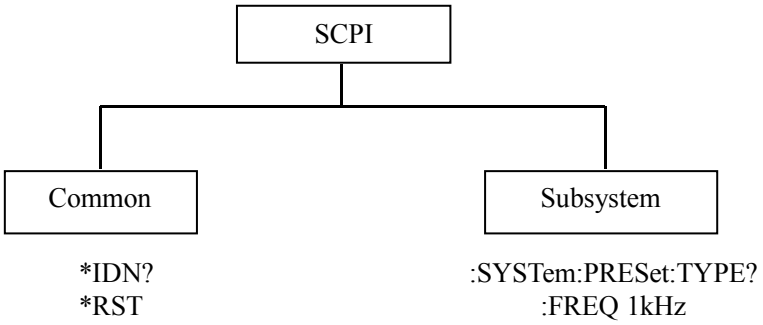


Figure 2.1 SCPI type

3) Instrument-specific Command Syntax

A typical command consists of keywords with colon as prefix. These keywords are followed by parameters. An example of syntax statement is shown below.

[:SENSe]:FREQuency[:CW|FIXed] MAXimum|MINimum

In the above example, the [:CW|FIXed] part of the command is closely followed by :FREQuency, and there is no space in the middle. The part closely following the [:LEVel]: The MINimum|MAXimum is the parameter part. There is a space between the command and the parameter. The description of other parts of the syntax expression is as shown in Table 2.2 and Table 2.3.

Table 2.2 Special Characters in the Command Syntax

Symbol	Meaning	Example
|	The vertical line between the keyword and the parameter represents a variety of options.	[:SENSe]:BANDwidth|BWIDth HIGH|LOWer BANDwidth and BWIDth are options, and HIGH and LOWer are options.
[]	Square brackets indicate that the included keywords or parameters are optional when they form a command. These implied keywords or parameters are executed even when they are ignored.	[:SENSe]:BANDwidth? SENSe is optional.

< >	The part inside the angle brackets can't be used literally in the command, instead, it represents the part that must be contained.	[:SENSe]:FREQency[:CW|FIXed] <val>[unit] In this command, <val> must be replaced by the actual frequency. [unit] means a unit that can be omitted. For example: FREQ 3.5GHz FREQ 3.5e+009
{ }	The part inside the braces indicates that the parameter is optional.	MEMory:TABLe:FREQuency <val>{,<val>} For example: MEM:TABL:FREQ 5e7

Table 2.3 Command Syntax

Character, Keyword and Syntax	**Example**
Capitalized characters represent the minimum character set required to execute the command.	[:SENSe]:FREQuency[:CW|FIXed]? FREQ is the short-format part of the command.
The lowercase character part of the command is optional; this flexible format is called "Flexible Listening". For more information, please refer to the section "Command parameter and response".	:FREQuency :FREQ,:FREQuency or :FREQUENCY, any of which is correct.
A colon between two command mnemonics moves the current path in the command tree downwards by one layer. For more information, please refer to the command path part in the section "Command tree".	:TRIGger:MODE? TRIGger is the topmost keyword of this command.
If the command contains multiple parameters, the adjacent parameters will be separated by commas. The parameter isn't a part of the command path, so it doesn't affect the path layer.	MEMory:TABLe:FREQuency <val>{,<val>}
The semicolon separates two adjacent commands but doesn't affect the current command path.	:FREQ 2.5GHZ;:POW 10DBM
Blank characters such as <space> or <tab> are usually ignored as long as they don't appear between the keywords or in the keyword. However, the command and parameter must be separated by a blank character, without affecting the current path.	:FREQ uency or :POWer :LEVel6.2 is not allowed. :LEVel and 6.2 must be separated by a space. i.e. :POWer:LEVel 6.2

The simplified syntax specification is as shown in Figure 2.2:

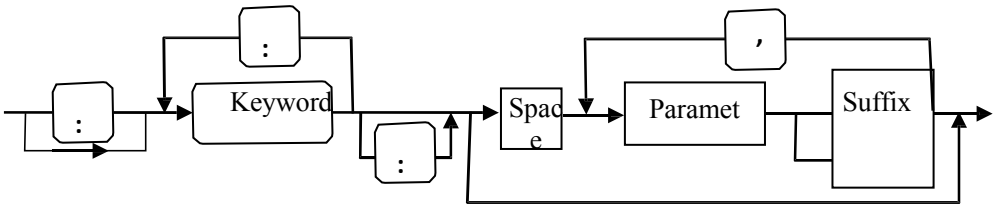


Figure 2.2 SCPI type

For example, the syntax expression of "[:SENSe[1]:FREQuency[:CW|FIXed] <Numeric Data>" can be expressed as follows.

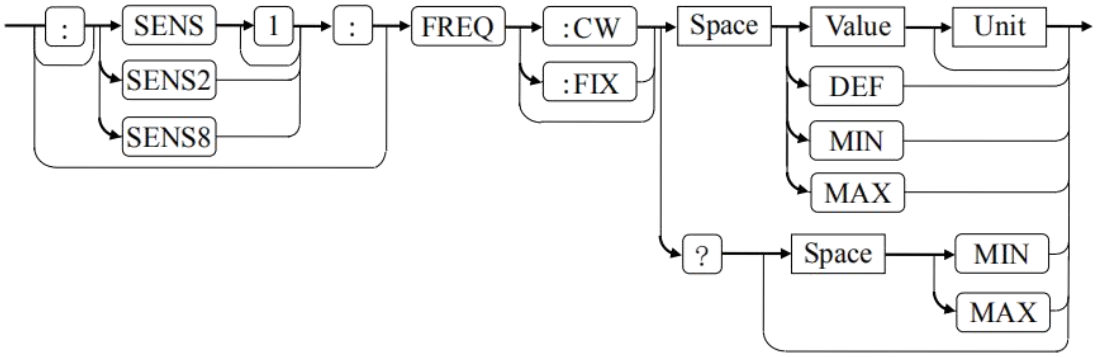


Figure 2.3 SCPI type

Remarks:

(1) The space can't be added in the above Figure unless otherwise indicated. There can be one or more spaces. If the unit is omitted, the standard unit of frequency and time is Hz and s respectively.

(2) The rounded rectangle represents the actual characters required for a keyword or a command, such as ":", ",", "?", "1"; the right-angle rectangle represents that it shall be replaced with actual characters, numbers, etc. If the "Value" can't appear in the command, it shall be replaced with the actual value, such as 5e+007.

(3) Regardless of the length format of the keyword, set the frequency to 50 MHz. The following several forms are displayed in the above Figure (only the short format of the keyword is taken. As there are a number of units of frequency, such as Hz, kHz, MHz, GHz, THz, they will not be given in details due to length limitations. It is only necessary to replace 5.0e+007 with corresponding units, such as 50 MHz, 5e+007 Hz, 0.05 GHz).

a) :SENS1:FREQ:CW 5.0e+007 No keyword will be omitted.
b) SENS1:FREQ:CW 5.0e+007 The ":" in front of the SENS1 will be omitted.
c) SENS:FREQ:CW 5.0e+007 The "1" will be omitted.
d) FREQ:CW 5.0e+007 The SENS will be omitted.
e) :SENS1:FREQ:FIX 5.0e+007 No keyword will be omitted.
f) SENS1:FREQ:FIX 5.0e+007 The ":" in front of the SENS1 will be omitted.
g) SENS:FREQ:FIX 5.0e+007 The "1" will be omitted.
h) FREQ:FIX 5.0e+007 The SENS will be omitted.
i) :SENS1:FREQ 5.0e+007 The CW or FIX will be omitted.
j) SENS1:FREQ 5.0e+007 The ":" and CW or FIX in front of the SENS1 will be omitted.
k) SENS:FREQ 5.0e+007 The "1" and CW or FIX will be omitted.
l) FREQ 5.0e+007 The SENS and CW or FIX will be omitted.

(4) The MIN and MAX can be used as parameters to set the command or to query the command. The DEF can only be used as a parameter to set the command. The specific values of the MIN, MAX and DEF are related to the instrument. The minimum, maximum and default frequencies are 1 kHz, 1 THz and 1 GHz respectively.

a) FREQ DEF Set the frequency to the default value.
b) FREQ? MAX Query the maximum settable frequency, and a value without units is returned.

(5) If consideration is given to the length form of the keyword but no consideration is given to the command with units, the command in the above Figure will have 1,632 forms totally. The user is unnecessary to care about all the forms, flexible use is enough. The following is a simple calculation, and the interested users can make calculation on

themselves.

a. Firstly, calculate the default condition of the SENS, and record it as N1. The FREQ/FREQuency has 2 forms, which is recorded as N11.

The CW/FIX/FIXed/omission has 4 forms, which is recorded as N12.

The parameters to set the command have 7 types, including value/DEF/DEFault/MIN/MINimum/MAX/ MAXimum.

The parameters to query the command have 5 types, including MIN/MINimum/MAX/MAXimum/omission, namely, there are 12 parameter forms, which is recorded as N13, and N1=N11*N12*N13=2*4*12=96.

b. After that, calculate the non-default condition of the SENS, and record it as N2. The first ":" has omission and no-omission types, which is recorded as N21.

The SENS/SENSe has 2 types. The affix has 4 types including 1/2/6/omission, namely, the SENS keyword has 2×4 types, 8 types totally, which is recorded as N22.

The SENS keyword and the following keyword form a combination relationship, N2=N21*N22=2*8*N1=16*N1.

c. Including N types of forms totally, and N=N1+N2=17*N1=1632.

4) Command Tree

Most remote control programming tasks involve instrument-specific commands. When such a command is parsed, the SCPI will use a structure similar to the file structure, and it is called as a command tree, as shown in Figure 2.4:

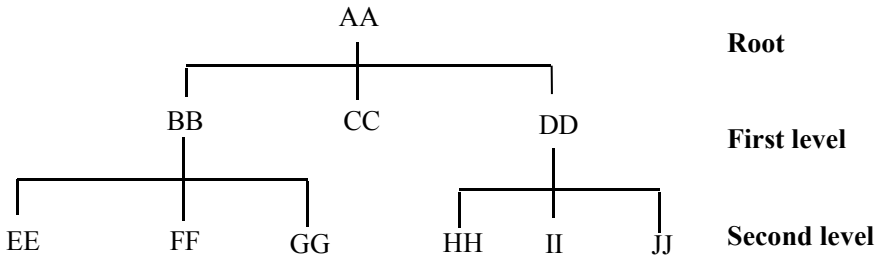


Figure2.4 Schematic Diagram of the Simplified Command Tree

The top command is root command, or simply "root". In the case of command parsing, the command at the next layer is reached by following a specific route based on the tree structure. For example::POWer:ALC:SOURce?, where, POWer stands for AA,: ALC stands for BB, :SOURce stands for GG, and the whole command path is (:AA:BB:GG).

A software module in the instrument software——**command interpreter** is used for parsing each received SCPI. The command interpreter breaks up the command into individual command element using a series of rules for identifying the command tree path. After the current command is parsed, the current command path remains unchanged. In this way, the subsequent commands can be parsed more quickly and efficiently because the same command keyword may appear in different paths. After the power-on*RST (reset) operation of the instrument, the current command path is reset as the root.

5) Command Parameter and Response

The SCPI defines different data formats in the use of the remote control and response messages to conform to the principles of "flexible listening" and "accurate speaking". For more information, please refer to IEEE 488.2. "Flexible Listening" means that the formats of commands and parameters are flexible.

For example, the instrument sets the frequency offset status command :FREQuency:OFFSet:STATe ON|OFF|1|0,

The following command formats are used to set the frequency offset function as "On":

:FREQuency:OFFSet:STATe ON, :FREQuency:OFFSet:STATe 1,

:FREQ:OFFS:STAT ON, :FREQ:OFFS:STAT 1.

Each parameter type corresponding to one or more response data types. During query, a numeric data will return a data type, and the response data is accurate. Strictly speaking, it is called as "**accurate speaking**".

For example, if you query the power state (:POWer:ALC:STATe?), when it is turned on, the response is always 1, regardless of whether you previously sent :POWer:ALC:STATe 1 or :POWer:ALC:STATe ON.

Table 2.4 SCPI parameters and response type

Parameter Type	Response Data Type
Numeric	Real or Integer
Extended Numeric	Integer
Discrete	Discrete
Boolean	Numeric Boolean
String	String
Block	Definite Length Block
	Indefinite Length Block
Non-decimal numeric	Hexadecimal
	Octal
	Binary

Numeric parameter

Numeric parameters can be used in both instrument-specific commands and common commands. It receives all common decimal systems including signs, decimal point and scientific notation. If a certain piece of equipment only receives a specified type of numeric parameter such as an integer, it will automatically round off the received numeric parameter.

Examples of numeric parameter:

0	No decimal point
100	Optional decimal point
1.23	With a sign bit
4.56e<space>3	space allowed after exponent marker e
-7.89E-01	exponent marker e may be upper or lower case
+256	leading + allowed
.5	The decimal point can be prefixed

Extended numeric parameter

Most measurements related to Instrument-specific commands use extended numeric parameters to specify the physical quantities. Extended numeric parameters receive all numeric parameters and additional special values. All extended numeric parameters receive MAXimum and MINimum as parameter values. Other special values, such as UP and DOWN are received by the instrument parsing capability. SCPI command table will list all valid parameters.

Note: extended numeric parameters are not applicable to common commands or STATus subsystem commands.

Examples of extended numeric parameters:

101	Numeric data
1.2 GHz	The GHz can be used as an exponent (E009)
200 MHz	The MHz can be used as an exponent (E006)
-100 mV	-100 millivolts
10DEG	10 degrees
MAXimum	Maximum valid setting
MINimum	Minimum valid setting
UP	Increase a step
DOWN	Decrease a step

Discrete parameter

When there are a finite number of parameter values to be set, discrete parameters are used for identification. A discrete parameter uses mnemonics to represent each valid setting. Same as the SCPI mnemonic, the discrete parameter mnemonic has long and short formats, and can be applied with mixed uppercase and lowercase letters.

The following example illustrates the combined use of discrete parameter and command.

:TRIGger[:SEQuence]:SOURce BUS|IMMediate|EXTernal

BUS	GPIB, LAN, RS-232 trigger
IMMediate	Immediate trigger
EXTernal	External trigger

Boolean parameter

Boolean parameters represent a single binary condition that is either true or false. There are only four possible representations for a Boolean parameter.

Samples of Boolean parameters:

ON	True
OFF	False
1	True
0	False

String parameter

A string parameter allows the ASCII string to be sent as a parameter. Single and double quotes are used as separators.

Examples of string parameter:

“This is Valid” “This is also Valid” “SO IS THIS”

Real response data

A large portion of measurement data are real. They are formatted as basic decimal notation or scientific notation. Most high-level remote control languages all support these two formats.

Examples of real response data:

1.23E+0

-1.0E+2

+1.0E+2 0.5E+0

0.23

-100.0

+100.0

0.5

Integer response data

The integer response data are a decimal expression of an integer with the sign bit. When the status register is queried, the integer response data will be mostly returned.

Examples of integer response data:

0 Optional sign bit

+100 Leading + allowed

-100 Leading - allowed

256 No decimal point

Discrete response data

The discrete response data and discrete parameters are basically the same. The main difference is that the discrete response data can only be returned in the short format with capitalized characters.

Examples of discrete response data:

INTernal Internal ALC mode

EXTernal External ALC mode

MMHead ALC mode being MMW source module

Numeric Boolean response data

The Boolean response data returns a binary value of 1 or 0.

String response data

The string response data and string parameters are the same. The main difference is that the string response data use double quotes rather than single quotes as the separator. The string response data can also be inserted with double quotes inside which there can be no characters.

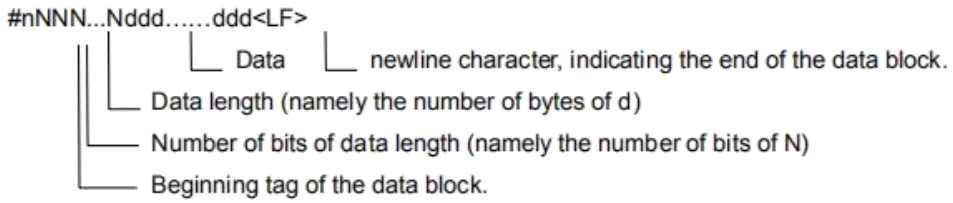
Examples of string response data:

“This is a string”

“one double quote inside brackets: (“”)”

Any data block

<Any Data Block> See section 7.7.6 of the IEEE 488.2.



For example: #42004…..<LF>, n = 4, N = 2,004.

6) Systems of Value in Commands

The value of the command can be entered in binary, decimal, hexadecimal or octal format. In the binary, hexadecimal, or octal format, a suitable identifier should be added in front of the value. In the decimal (default) format, an identifier isn't required. When the value without an indicator is entered, the equipment will ensure that it is entered in decimal format. The identifiers required in all formats are listed as follows:

- #B indicates that this digit is a binary value.
- #H indicates that this digit is a hexadecimal value.
- #Q indicates an octal number.

The representations of the decimal value 45 in the SCPI are given as follows:

#B101101

#H2D

#Q55

The following example shows setting of the RF output power as 10 dBm (or the value equivalent to the current selected unit including DBUV or DBUVEMF) with the hexadecimal value 000A.

:POW #H000A

When using a non-decimal format, a measurement unit, such as dBm or mV, is not used together with the value.

7) Command Line Structure

A command line may contain a number of SCPIs. To indicate the end of the current command line, the following methods can be used:

- Newline;
- Newline and EOI;
- EOI and the last data byte.

Commands in command line are separated by semicolons, and commands for different subsystems begin with a colon. For example:

MMEM:COPY "Test1", "MeasurementXY";:HCOP:ITEM ALL

The command line contains two commands of which the first one belongs to the MMEM subsystem and the second one belongs to the HCOP subsystem. If the adjacent commands belong to the same subsystem, the command path will be partially repeated and the command can be abbreviated. For example: For example:

HCOP:ITEM ALL;:HCOP:IMM

The command line contains two commands both of which belong to the HCOP subsystem of first level. Therefore, the second command can begin with the subordinate to HCOP and may not begin with a colon, which can be abbreviated to the following command line:

HCOP:ITEM ALL;:HCOP:IMM

2.1.4 Command Sequence and Synchronization

IEEE488.2 defines the difference between overlapped commands and sequential commands:

- Sequential commands are sequences of commands that are executed continuously. Generally, the execution of each command is faster;
- Overlapped commands indicate that the previous command is not executed automatically before the next command is executed. Normally overlapped commands take longer to process and allows the program to process other events synchronously.

Even if multiple commands are set in a command line, they are not necessarily executed in the order in which they are received. In order to ensure that the commands are executed in a certain order, each command must be sent as a separate command line.

Example: Command line contains set and query commands

If multiple commands in a command line contain query commands, the query result is unpredictable. The following command returns a fixed value:

:FREQ:STAR 1GHZ;SPAN 100;:FREQ:STAR?

Returned value: 1,000,000,000 (1 GHz)

The following command returns an unfixed value:

:FREQ:STAR 1GHz;STAR?;SPAN 1000000

The returned result may be the current frequency start value, because the host program will delay execution of the command. If the host program receives and executes the command, the returned result may also be 1 GHz.

Note

Setting command and query command are sent separately

General rule: In order to ensure the correctness of the returned result from the query command, the setting command and the query command shall be sent in different program control messages.

Preventing overlapping execution of the command

In order to prevent the overlapped execution of commands, multiple threads or commands: *OPC, *OPC? or *WAI can be used. These three commands can be executed only after the hardware is set. While programming, the computer can be forced to wait for some time to synchronize certain events. The details are separately described below:

- **Controller program uses multiple threads**

Multi threads are used to wait for completion of the command and achieve synchronization of GUI and program control, that is, a single thread waits for completion of *OPC?, without impeding the execution of the GUI or remote control thread.

- **The methods for use of three commands during synchronous execution are given in the table below:**

Table 2.5 Command Syntax

Method	Actions to be Executed	Programming Method
*OPC	Set the operation completion bit is set.	Set ESE BIT0; Set SRE BIT5; Send the overlapped command and *OPC; Wait for Service Request (SRQ) signal; SRQ represents the completion of execution of the overlapped command
*OPC?	Stop executing the current command until the value 1 is returned. The command is returned only when the operation completion bit in the ESR is set, which indicates that the previous command is processed.	Terminate the processing of the current command before executing other commands. Send this command directly after the current command.

If the processing time of the overlapped command is short, the command *WAI or *OPC can be used after use of the overlapped command to achieve command synchronization. In order to synchronously execute other tasks when the computer or instrument is waiting for the completion of execution of overlapped commands, the following synchronization technologies can be adopted:

- **OPC and service request**

1) Set the ESE OPC mask bit (bit0): *ESE 1;
2) Set the SRE bit5: *SRE 32 and enable ESB service request;
3) Send the overlapped command and *OPC;
4) Wait for the service request signal.

SRQ represents the completion of execution of the overlapped command

- **OPC? and service request**

1) Set the SRE bit4: *SRE 16 enables the MAV service request;
2) Send overlapped commands and *OPC? ;
3) Wait for the service request signal.

SRQ represents the completion of execution of the overlapped command

- **Event Status Register (ESE)**

1) Set the ESE OPC mask bit (bit0): *ESE 1;
2) Send the overlapped command only and do not send *OPC, *OPC or *WAI;
3) Send "*OPC;*ESR?" in the timer for cyclic query of completion status of operation.

If the return value (LSB) is equal to 1, this indicates that the overlap command has been executed.

- ***OPC? and short timeout**

1) Send the overlapped command only and do not send *OPC, *OPC or *WAI;
2) Send "<short timeout>; *OPC?" in the timer, so as to query the completion status of operation circularly;
3) If the return value (LSB) is equal to 1, this indicates that the overlap command has been executed. During timeout, during operation;
4) Reset the timeout value to the old value;
5) Send the command "SYSTem:ERRor?" clear the error queue, and delete the "-410 Query Interruption" information.

If the return value (LSB) is equal to 1, this indicates that the overlap command has been executed.

2.1.5 Status Reporting System

The status reporting system will save all operation status information of the current instrument, including error information. Such information is storedin the status register and error queue respectively and can be queried through the remote interface.

2.1.5.1 Structure of the status register

The register classification is described as follows:

1) STB, SRE

Status Byte (STB) register and its associated mask register, Service Request Enable (SRE) register, constitute the

top-level register of the status reporting system. The STB saves the general working status of the instrument by collecting low-level register information.

2) ESR, SCPI status register

STB receives the information of the following registers:

- The value of Event Status Register (ESR) and Event Status Enable (ESE) mask register.
- The SCPI status registers include: STATus:OPERation and STATus:QUEStionable registers(SCPI definition) which contain the specific operating information of the instrument. All the SCPI status registers shall have the same internal structure (please refer to Section 2.1.5.2 " Structure of SCPI status register" in the Programming Manual).

3) IST, PPE

Similar to the SRQ, an individual bit of the IST mark ("Individual STatus") is a combination of all statuses of the instrument. The associated parallel query enable register (PPE) determines which data bits of the STB act on the IST mark.

1) Output buffer

The output buffer stores the message returned by the instrument to the controller. It doesn't belong to the status reporting system but determines the value of the MAV bit of STB.

For details of above register descriptions, please refer to "2.1.6 Status reporting system".

Note

SRE, ESE

The SRE can be used as an enable part of the STB. Similarly, the ESE can be used as an enable part of the ESR.

2.1.5.2 Structure of SCPI Status Register

Each standard SCPI register consists of 5 parts. Each part contains 16 data bits and is functionally independent. For example, each hardware status will be assigned with a data bit, and it is valid for all 5 parts of the register. If the Bit15 is set to 0, it means that the value of the register is a positive integer.

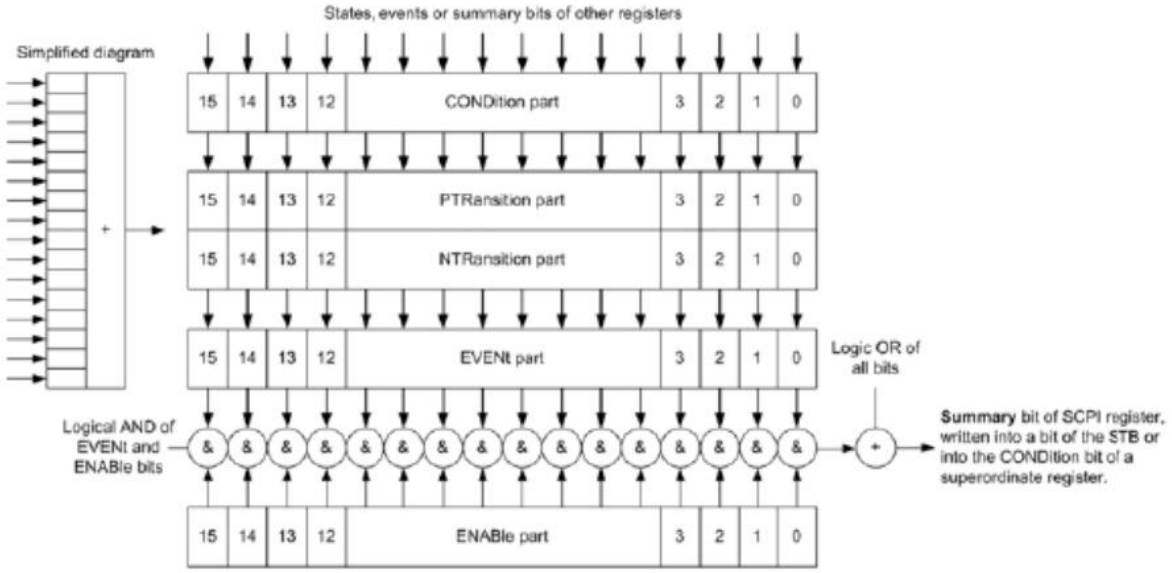


Fig.2.5 Structure of the status register

The above figure shows that the status register is composed of 5 parts, which are described as follows:

- **Condition register**

This part will be directly written by hardware or low-level register digit, which will reflect the current working state of the instrument. This register is read-only and cannot be written. Reading will not clear any value.

- **Positive and negative jump register**

The two jump registers define the status jump bit of the condition register saved in the event register.

The positive jump register is similar to a jump filter. When a data bit of the condition register changes to 1 from 0, relevant PTR bit will determine whether the event bit is set to 1. The description is as follows:

—PTR bit=1: The event bit will be set.

—PTR bit=0: No event bit will be set.

This positive jump register is read-write, and no value will be cleared after reading.

The negative jump register is similar to a jump filter. When a data bit of the condition register changes to 0 from 1, relevant NTR bit will determine whether the event bit is set to 1. The description is as follows:

—NTR bit=1: The event bit will be set.

—NTR bit=0: No event bit will be set.

This positive jump register is read-write, and no value will be cleared after reading.

- **Event register**

This part indicates whether the event occurs after the last reading, and whether the content of the condition register is saved. It only represents the event transmitted by the jump register, which can only be changed by the instrument and read by the user. The value will be cleared after reading. The value of this part is equal to the value of whole register generally.

- **Enable register**

This part determines whether the associated event bit acts on the final data sum. The data bit of each enable part has a And relation with the associated enable bit. The logical operation result of this part has a OR relation with the data sum bit.

—Enable bit = 0: The associated event bit does not act on the data sum.

—Enable bit=1: The associated event bit acts on the data sum

This part is read-wrote, and no value will be cleared after reading.

➢ **Data bit sum**

The data sum bit of each register consists of event and enable parts. The result gets into the condition part of the high level register. The instrument automatically generates data sum bit for each register so that events can cause different levels of service requests.

2.1.5.3 Status Register Description

The status registers will be introduced in details as follows.

1) Status byte (STB) and service request enable register (SRE)

The IEEE488.2 defines the status byte (STB) that reflects the rough status of the instrument by collecting the information of the low-level register. The bit6 is equal to the data sum of other status byte bits. The result after comparing the status byte with the condition part of the SCPI register can be assumed to be the highest level in the SCPI level. The common command "*STB?" or the serial query can read the status byte value.

The status byte is connected with the service request enable register (SRE). Each data bit of the status byte corresponds to one bit in the SRE. The SRE bit6 is ignored. If one data bit in the SRE is set and the associated STB bit changes to 1 from 0, a service request (SRQ) will be generated. The common command "*SRE" is used to set the SRE, and the common command "*SRE? " is used to read the SRE. The status byte is described in the following Table 2.6 Description of the Status Byte.

Table 2.6 Description of the Status Byte

Data Bit	Meaning
0..1	Not used.
2	The error queue is non-null Set the bit if a new error is inserted into the error queue. If the associated SRE bit enables the bit and a a new error is generated in the error queue, a service request will be generated to identify the error and query the error information. This method effectively reduces the error in remote control.
3	Data sum bit of the status query register The bit can be set if the event bit of the status query register and the associated enable bit are set to 1. The bit represents a queriable status of the instrument, and the specific status information of the instrument can be obtained by querying the status query register of the status register.
4	MAV bit (message available) Set the bit if the output queue information is readable. Use the bit when the controller queries the instrument information.
5	ESB bit Data sum bit of the event status register. The bit can be set if one bit of the event status register is set and the enable event enables the corresponding bit in the register. If the position bit is 1, it means that the instrument has a severe error, and the specific error information can be obtained by querying the event status register.
6	MSS bit (master status summary bit) Set the bit if the instrument triggers the service request.

7	Data sum bit of the operation status register The bit can be set if the event bit of the operation status register and the corresponding enable bit are set to 1. This bit indicates that the instrument executes an operation, and the specific operation type can be obtained by querying the operation status register.

2) IST mark and parallel query enable register (PPE)

The IST identifies the combination of the overall status of the instrument with a separate data bit. This flag can be obtained by parallel query or by sending the command "*IST?". The associated parallel query enable register (PPE) determines which data bits of the STB act on the IST mark. The STB data bits have the And relation with the PPE data bits, and the usage of bit6 is opposite to that in the SRE. The IST flag is equal to the Or value of all results. Set and read the PPE through the command "*PRE" and the command "*PRE?" respectively.

3) Event status register (ESR) and event status enable register (ESE)

The IEEE488.2 defines the ESR. The event status register (ESR) can be read through the command"*ESR?". The ESE is an enable part of the SCPI register. If one position is set to 1 and one data bit in the responsive ESR changes to 1 from 0, the ESB bit of the STB will be set to 1. Set and read the ESE through the command "*ESE" and the command "*ESE?" respectively.

Table 2.7 Description of the Event Status Byte

Data Bit	Meaning
0	Operation completed The bit can be set when the the previous commands have been executed and the command *OPC has been received.
1	Not used.
2	Query error This bit is set if the controller reads the instrument data without sending a query command or sends a new command without reading the query data. It means that a wrong query is generated and the query can't be executed.
3	Instrument error Set the bit if an instrument error is generated. Error code range: -300 to -399, or positive error code. For details of specific error information, query relevant information in the error queue.
4	Execution error Set the bit if a command with correct syntax is received but can't be executed. In addition, an error with the code within the range of -200 to -300 is generated in the error queue.
5	Command error Set the bit if the received command has a syntax error. Error code range: -100 to -200. For details of specific error information, query relevant information in the error queue.
6	User request Set the bit if the instrument is switched to the manual control mode.
7	Power on Set the bit when the instrument is powered on.

4) Status: Operation register

Status: The operation register includes the current instrument operation information, and previously executed operation information. The value of the operation register can be read through the command "STATus:OPERation:CONDition?" or "STATus: OPERation[: EVENt]?". The description of the status register is as

shown in the following Table 2.8.

Table 2.8 Status: Description of operation register

Bit	Value	Definition
0	1	Not used
1	2	Calibration state
2	4	Reserved
3	8	Reserved
4-14	-	Not used
15	-	It is always 0

5) Status: Questionable register

The register includes the status of the instrument not conforming to requirements of the specification. The value of the register can be queried through the command “STAT:QUES:COND” or “STAT:QUES:EVEN”. The description of the status register is as shown in the following Table 2.9.

Note

Query register

Status: The questionable register gathers the information (for example: bit2 gathers all time-related information) of all the low-level sub-registers. As each channel corresponds to the independent sub-register, if one status bit indication of the questionable register has an error, it is necessary to trace back in the channel sub-register and check the specific error root. The status of the sub-register state to be queried belongs to the currently selected channel by default.

Table 2.9 Status: Description of questionable register

Bit	Value	Definition
0-2	-	Not used
3	8	Power summary
4-7	-	Not used
8	256	Calibration summary
9	512	Power-on self-test
10-14	-	Not used
15	-	It is always 0

2.1.5.4 Application of status reporting system

The status reporting system is used to monitor the status of one or more instruments in the test system. To correctly realize the function of the status reporting system, the controller in the test system must receive and evaluate the information of all instruments. The standard methods applied include:

1) Instrument initiated Service Request (SRQ);
2) Series query of all the instruments in the bus system, initiated by the controller in the system, in order to find the initiator and the cause of service request;

3) Perform parallel query of all instruments;

4) Query of the status of specific instrument by remote control command;

5) Query of the error queue.

1) Service request

In some cases, the instrument will send a service request (SRQ) to the controller to obtain the controller's service, and then the controller will initiate an interruption to enter the corresponding interruption handler. As shown in Figure 2.5, an SRQ is usually initiated by one or more status bytes and the 2nd, 3rd, 4th, 5th or 7th bits of the associated enable register (SRE). These data bits are composed of the advanced register, error queue, or output buffer area further. In order to use all the service requests as much as possible, all data bits of the enable registers SRE and ESE shall be set to 1.

Example: Generate the SRQ signal with the command *OPC when scanning is completed.

a) Recall the write function InstrWrite and write command "*ESE 1", and set the ESE bit0 (operation is completed).

b) Recall the write function InstrWrite and write command "*ESE 32", and set the SRE bit5 (ESB).

c) Recall the write function InstrWrite and write command "*INIT;*OPC", and generate the SRQ signal after operation is completed.

The instrument will generate an SRQ after settings are completed.

The SRQ can only be initiated by the instrument, and the controller program shall allow sending the service request to it when the instrument has an error, and it will be processed by a special interruption handler.

2) Series query

Similar to the command *STB, the series query can be used to query the status byte of the instrument. The series query adopts the interface message mode, therefore, the query is fast. The IEEE 488.2 defines the specific series query method. This method is mainly used to quickly obtain the status of one or more instruments connected with the controller in the test system.

3) Parallel query

The controller can send an information bit to the USB cable through a command, and query 8 instruments in the test system. The data configured on the USB cable of the instrument is logic "0" or "1". Except that the SRE register determines the conditions generating the SRQ, perform parallel query to check AND operation of data bit of the enable register (PPE) and the STB register. The result will be made as the response result and sent to the parallel query controller through OR operation and bit reverse, and it can also be obtained through the command *IST.

During parallel query, set the instrument to the parallel query status through the command PPC firstly, and this command will allocate a USB cable to the instrument and determine whether the data bit is reversed in response. Use the PPE register during execution of the parallel query. The parallel query is mainly used to quickly position which instrument sends the service request by the controller. Accordingly, the same value shall be set for the registers SRE and PPE.

4) Instrument status query

Query each part of the status register through following two commands:

- The commands *ESR?, *IDN?, *IST?, *STB? can be used to query the advanced register;
- The status system command can be used to query the SCPI register (for example: STATus: QUEStionable…).

The returned value of the queried register is usually in decimal format, which is used by the controller program for detection. In order to obtain a more detailed description of the SRQ cause, the parallel query will be carried out after the SRQ generally.

Description of response data bit

The STB and ESR registers include 8 bits, and the SCPI register includes 16 bits. The returned value of the queried status register is in decimal format. The decimal value is equal to the sum of the data bits and their own weights after operation.

The relationship between the data bit and its weight is as shown in the following Figure:

Fig.2.6 Diagram of Relationship Between the Data Bit and Its Weight

5) Error queue

Each error status of the instrument corresponds to an entry in the error queue, including the specific error information text, which can be viewed through the error log or queried through the SCPI SYSTem:ERRor[:NEXT]? or SYSTem:ERRor:ALL?. If there is no error in the error queue, the query will return 0, “No Error”.

As the obtained error cause description is more accurate than the status register, the error queue shall be queried in the controller service request handler. The error queue shall be frequently queried especially during the controller program test stage, so as to clarify the error command record sent to the instrument by the controller.

2.1.5.5 Reset Atatus Reporting System

The following list shows the commands and events for resetting the status report system. Except for the commands *RST and SYSTem:PRESet, other commands will not change the function setting of the instrument. Similarly, the DCL will not change setting status of the instrument. The specific description is as shown in the following table:

Table 2.10 Reset status reporting system

Event / Function	Power On/Off (Power-on status cleared)		DCL, SDC (Instrument cleared, selected instrument cleared)	*RST or SYSTem: PRESet	STATus: PRESet	*CLS
	0	1				
Clearing STB, ESR	—	Yes	—	—	—	Yes
Clearing SRE, ESE	—	Yes	—	—	—	—
Clearing PPE	—	Yes	—	—	—	—
Clearing the event part of the register	—	Yes	—	—	—	Yes
Clearing the enable part of the operation and questionable registers. Filling 1 in the enable part of other registers.	—	Yes	—	—	Yes	—

Filling 1 in the positive jump part. Clearing the negative jump part.	—	Yes	—	—	Yes	—
Clearing the error queue	Yes	Yes	—	—	—	Yes
Clearing the output buffer area	Yes	Yes	Yes	—	—	—
Clearing the command processing and input buffer area	Yes	Yes	Yes	—	—	—

2.1.6 Programming Considerations

1) Please initialize the instrument status before changing the settings

When setting the instrument by remote control, the status of the instrument (such as sending "*RST") shall be initialized firstly, and then the required status setting shall be realized.

2) Command sequence

In general, the setting and query commands should be sent separately; otherwise the returned value of the query command will change according to the current instrument operation sequence.

3) Failure response

The service request can only be initiated by the instrument itself. The controller program in the test system should instruct the instrument to initiate a service request when an error occurs, and then enter the corresponding interrupt service routine for processing.

4) Error queue

Each time the controller program processes a service request, the error queue rather than the status register of the instrument should be queried to obtain a more accurate error reason. The error queue should be frequently queried to obtain the wrong command sent by the controller to the instrument especially during testing of the controller program.

2.2 Remote Interface and Its Configuration

The programming interface supported by this instrument is: USB. This interface can be used together with the I/O library and programming language for remote control of the instrument.

No remote configuration is required for S87230 front panel. USB interface conforms to the USBTMC (USB Test and Measurement Class) protocol, and it is a set of enumerable equipment (namely, it can be found by viFindRsrc function of the VISA library). The Vendor ID (VID) is 0x04B4 and Product ID (PID) is 0x1004. The serial number is as shown on each sensor.

Take S87231 as an example, to realize its remote control through the VISA library, the user shall install the NI or Keysight VISA library, and correctly install the USBTMC drive.

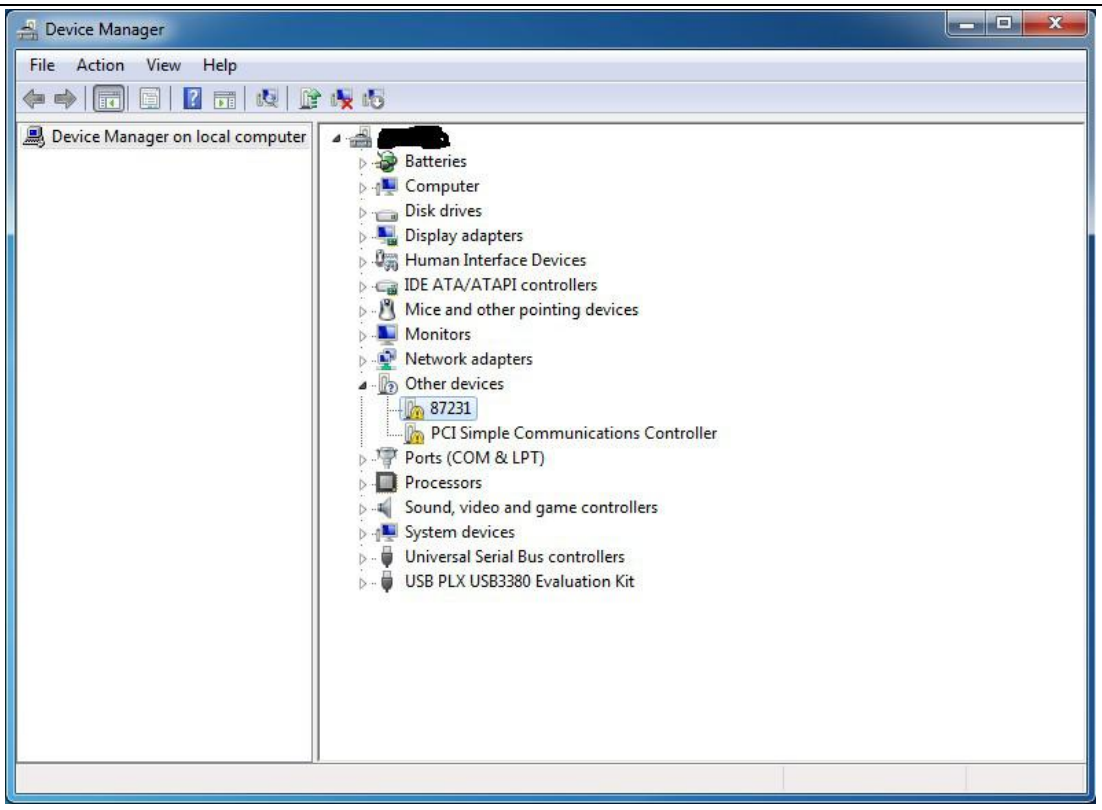


Fig.2.6 Device Manger 1

If the S87230 probe is used for the first time, the system will first automatically detect the device named “87231”, as shown in Fig. 2.6 (if it can‟ t be determined, you can check the “Hardware Id” attribute value in the device details, and the Vender id (VID) is 0x04B4 and Product id (PID) is0x1004). The steps are as follows:

Step 1: Update the drive, right click “87231” device name, and select “Update Driver Software”. After that, the “Update Driver Software-87231” dialog box will pop up. The user can select the “Browse my computer for driver software” option below, as shown in Fig. 2.7:

Fig. 2.7 Update Driver Software 1

Step 2: After that, continue to select the "Let me pick from a list of device drivers on my computer" option below the the pop-up dialog box, as shown in Fig.2.8:

Fig. 2.8 Update Driver Software 2

Step 3: Finally select "USB Test and Measurement Device (IVI)" device and click "Next", as shown in Fig. 2.9:

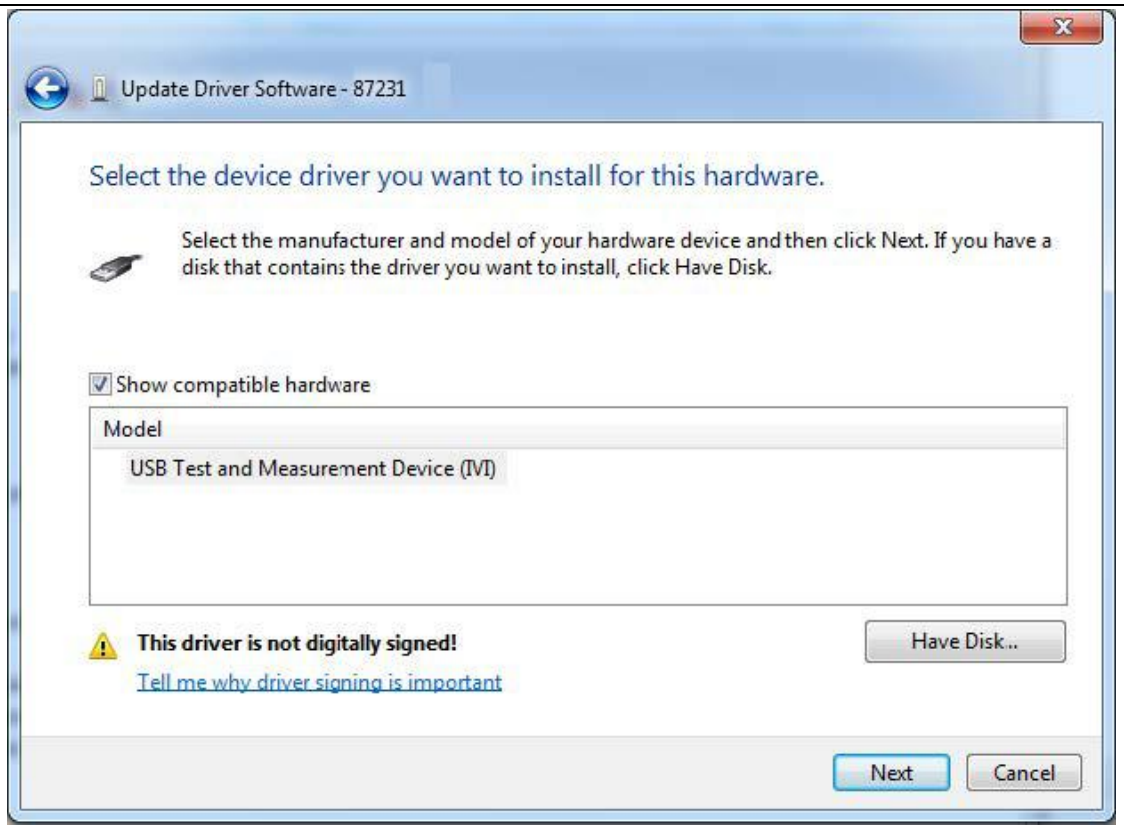


Fig. 2.9 Update Driver Software 3

After the driver update is completed, the device manager will display "USB Test and Measurement Devices", and the responsive USBTMC device will be detected in its list, as shown in Fig. 2.10:

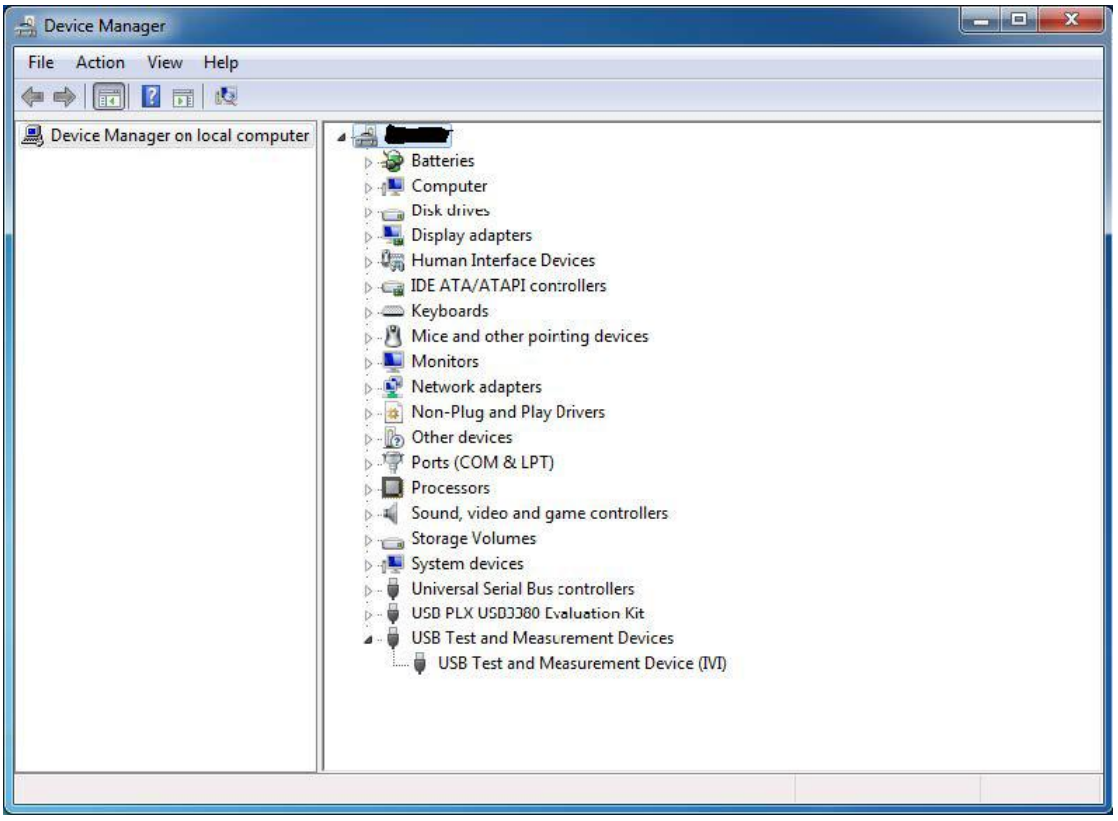


Fig. 2.10 Device Manger 2

2.3 I/O Library

2.3.1 Overview of I/O Library

As a library of software programs pre-written for the instrument, the I/O library is called an instrument driver. It is considered as the intermediate layer of the software between the computer and the instrument hardware equipment, composed of function library, utility program and tool kit, and used as a software code module set that corresponds to a planned operation, e.g. configuring, reading from, writing to or triggering the instrument. It resides in the computer as the bridge and link between the computer and the instrument and provides a easily programmed high-level modular library so that the user no longer needs to learn complex low-level programming protocols specific to an instrument. The instrument driver is the key to rapid development and test of measurement applications.

From the aspect of function, a general instrument driver usually consists of a functional body, an interactive developer interface, a program developer interface, a subprogram interface and an I/O interface as shown in Fig. 2.11.

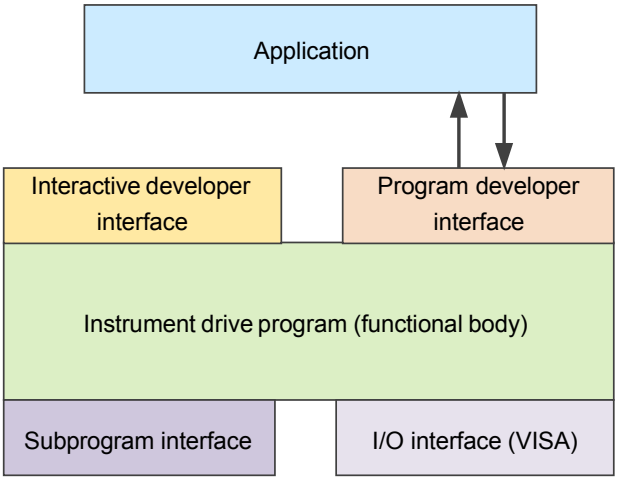


Fig. 2.11 Structure Model of the Instrument Driver

The detailed description is given as follows:

1) Functional body. It is the main functional part of the instrument driver and can be understood as the framework program of the instrument driver.
2) Interactive developer interface. For user's convenience, a graphical interactive developer interface is generally provided in the application development environment that supports the development of the instrument driver. For example, the function panel in Labwindows/CVI is an interactive developer interface. In the function panel, each parameter of the instrument driver function is represented by a graphical control.
3) Program developer interface. It is a software interface for recalling of the instrument driver function by the application, such as the dynamic link library file .dll of the instrument driver of the Windows system.
4) I/O interface. It is used to complete the actual communication between the instrument driver and the instrument. Bus-specific I/O software (such as GPIB and USB) and also common standard I/O software (VISA I/O) across multiple buses can be used.
5) Subroutine interface. It is a software interface for the instrument driver to access other support libraries including database and FFT function. When the instrument driver needs to recall other

software modules, operating systems, program code libraries and analysis function libraries to complete its task, the subprogram interface will be used.

2.3.2 Installation and Configuration of I/O Library

With the development of the test field application from the traditional instrument to the virtual instrument, the instrument driver has experienced different development processes in order to solve the instrument interchangeability and test program reusability of the automatic test system. Currently, the IVI (Interchangeable Virtual Instruments) driver is popularly applied. Based on the IVI specification, a new instrument programming interface is defined, the class driver and VPP architecture are inserted onto the VISA so that the test application is completely independent of the instrument hardware, and unique instrument simulation, range detection and status buffer functions are added, which improves the system operation efficiency and truly achieves the instrument interchange.

The IVI driver comes in two types: IVI-C and IVI-COM. IVI-COM is based on Microsoft Component Object Model (COM) technology in the form of COM API; IVI-C is based on ANSI C in the form of C API. These two types of drivers are designed according to the instrument defined by the IVI specification, and their application development environments are the same, including Visual Studio, Visual Basic, Keysight VEE, LabVIEW, CVI/LabWindows.

Currently, it is necessary to provide two types of drivers in order to meet the demands of different users in different development environments. The IVI driver of the instrument is developed based on Nimbus Driver Studio so that the IVI-COM and IVI-C drivers and program installation packages are generated directly. For details about installation and configuration, please refer to the accompanied documents of your selected control card and I/O library.

The installed IVI driver is divided into an IVI intrinsic functional group and a instrument class functional group (a basic functional group and an extended functional group). For details about functional classification, functions and attributes, please refer to the accompanied help document of the driver.

Note

Configuration of ports and installation of I/O library

When a computer is to be used to control the instrument, please confirm that the necessary ports and I/O library have been properly installed and configured.

Note

Use of I/O library

Once installed, the attached IVI-COM/C driver installation package will automatically install the driver function panel, help documents, and sample programs of the driver functions to facilitate the users to develop and integrate the program control functions

2.4 Zeroing

The zeroing is used to deduct the channel noise. No power signal can be sent to the sensor before zero. When will zeroing be carried out? It is recommended to zero the instrument in the following conditions:

- When the temperature change is larger than 5℃;
- When replacing the power sensor;
- After 24 h;
- Before measuring the low power signal. For example, when measuring the signal 10 dB lower than the minimum power specified by the power sensor.

The relevant SCPIs are:

CALibration[1]:ZERO

For example, zero:

CAL:ZERO

2.5 Measurement

The measurement can be configured as forms including absolute power measurement and relative power measurement. For details, please refer to the “Measurement of the Subsystem”.

2.6 Use of FDO Table

How to use FDO table? The FDO table is used to compensate and measure the frequency response during establishment.

2.6.1 Overview

Enable or disable FDO table with SENSe:CORRection:CSET:STATe. When FDO table is enabled, it will provide a quick method to compensate for the frequency response in the test system. It is important to note that when FDO table is enabled, the frequency offset is an “additional” sensor frequency response, namely, the inherent frequency response (saved in the sensor EEPROM) of the sensor shall also be considered. The sensor can save 1 FDO table, up to 80 frequency points for each.

Method to use FDO table:

a. Entering in FDO table
b. Enabling FDO table
c. Specifying frequency
d. Carrying out measurement

2.6.2 Entering in FDO Table

a. Entering steps

- Enter the frequency list :MEMory:TABle:FREQuency <Frequency 1>{, <Frequency i>}. Such as 50 MHz, 1 GHz, 10 GHz, 40 GHz.
- Enter the corresponding offset factor of the frequency list: MEMory: TABle:GAIN <Factor 1>{, <Factor i>}. For example, 100, 98.8, 101.2, 110.8 respectively correspond to 50 MHz, 1 GHz, 10 GHz, 40 GHz offset factors.

b. List name of FDO table: MEMory: CATalog:TABLe?. For specific information, see description of the command.

c. Rename of FDO table: MEMory:TABLe:MOVE <Former Name>, <Target Name>.

d. Query data in FDO table. For example, query data in the FDO table.

- Query number of frequency points in the FDO table :MEMory:TABLe:FREQuency:POINTs?
- Query the frequency list in the FDO table :MEMory:TABLe:FREQuency?
- Query number of factor points in the FDO table :MEMory:TABLe:GAIN[:MAGNitude]:POINTs?
- Query the factor list in the FDO table :MEMory:TABLe:GAIN[:MAGNitude]?

e. Modify data of FDO table: See 5.2 Entering in FDO table.

f. Description

- The frequency list must be in ascending order.
- The effective suffix of the frequency list includes Hz, kHz, MHz, GHz. The default value is Hz.
- Ensure that the frequency list is within the effective frequency range of the sensor.

2.6.3 Selecting FDO Table

SENSe:CORRection:CSET:STATe ON

2.6.4 Enabling FDO Table

[SENSe[1]|SENSe:CORRection:CSET:STATe ON

2.6.5 Measurement Application

- ABOR
- CONF:POW:AC DEF,1,(@1)
- SENS:CORR:CSET:SEL "MyFdo0"
- SENS:CORR:CSET:STAT ON
- SENS:FREQ 5GHz
- INIT1
- FETC?

2.6.6 Specific Application

a. If the frequency list of FDO table is 500 MHz, 1 GHz, 11 GHz. The offset factor list is 100, 10, 10. Set the signal frequency to Freq and the calculated offset factor to Gain.

b. If the Freq is out of the range of FDO table, the frequency offset value of the highest or the lowest frequency points in FDO table shall be used. In case of 18 GHz, use the corresponding offset factor 102 of the maximum frequency point 10 GHz. In case of 50 MHz, use the corresponding offset factor 100 of the minimum frequency point 500 MHz.

c. If the Freq is within the effective range of the frequency list but it is between two frequencies (Freq1, Freq2), such as 5 GHz, the offset factor will be obtained by two linear interpolations. Set the corresponding offset factors of the Freq1 and Freq2 to Gain1 and Gain2 respectively. The Gain can be calculated to be 50 by employing the following formula

$$Gain = Gain1 + \frac{Freq - Freq1}{Freq2 - Freq1} \times (Gain2 - Gain1)$$

d. If the power before using FDO table is 1.000 mW (recorded as Pwr0) and the final display power is recorded as Pwr, the Pwr＝Pwr0/Gain/100 = 2.000 mW.

2.7 Setting of Display Resolution

It is not supported temporarily.

2.8 Setting of Average

The sensor range of S87230 is classified into high and low ranges, which can be set as automatic range and manual range modes. If the measured power level is unclear, automatic range can be used.

Set the SCPI of the automatic range status:

SENSe:POWer:AC:RANGe:AUTO <Switch>

For example, set the sensor to automatic range on: SENS:POW:AC:RANG:AUTO 1

For example, set the sensor to automatic range off: SENS:POW:AC:RANG:AUTO 0

Set the SCPI of keeping the range:

SENSe:POWer:AC:RANGe <Range>

The <Range> value is as follows: 0 represents low range and 1 represents high range.

For example, set the sensor to high range

SENS:POW:AC:RANG 1

2.9 Setting of Range

The S87230 has a digital filter for average power reading. The range of the average number of times is 1 - 1,024. If the average status is enabled, the measurement time will be increased.

The S87230 CW power sensor can be set to the automatic average status, namely, set different average numbers of times according to different power levels and the display resolutions. Generally, the lower power will bring the larger average number of times.

Relevant SCPIs:

SENSe:AVERage[:STATe] <Switch> Set the average switch

For example, turn on average switch of the sensor:

SENS:AVER 1

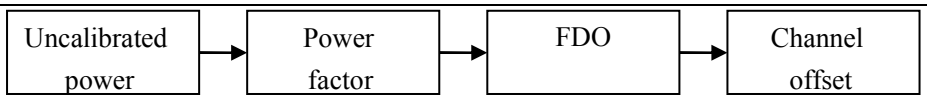
Turn off average switch of the sensor. AVER 0

SENSe:AVERage:COUNt:AUTO <Switch> Set the automatic average switch

SENSe:AVERage:COUNt <Average Number of Times> Set average number of times

2.10 Setting of Offset

The S87230 can compensate for signal attenuation or gain in the test equipment (such as compensating for 20 dB attenuator). The channel offset is added to the power measurement before its display.



The relevant SCPIs are:

SENSe:CORRection:GAIN2 <Offset Value> Set the channel offset value

SENSe:CORRection:GAIN:STATe <Switch> Set the channel offset switch

For example, set the sensor channel offset to 3 dB.

SENS:CORR:GAIN2 3

2.11 Setting of Measurement Limit

Verify if the measured power is out of the given range by setting the measurement limit. The relevant SCPIs are:

:CALCulate:LIMit:LOWer[:DATA] <Lower Limit> Set the lower limit

:CALCulate:LIMit:UPPer[:DATA] <Upper Limit> Set the upper limit

:CALCulate:LIMit:STATe <Switch> Set the limit detection switch

:CALCulate:LIMit:FAIL? Query if it is out of the limit

:CALCulate:LIMit:FCOunt? Query if it is out of the number of limits (FCO)

CALCulate:LIMit:CLEar[:IMMediate] Reset failure count (FCO).

2.12 Status Reporting

The status report is used to detect power error information, operation status, questionable status, etc. The status report will be reported level by level. For example, the calibration operation will be reported to the operation status and the operation status will be reported to the status word.

For example, query if the sensor is being calibrated, if the sensor calibration is completed, if zero and calibration error occurs, if zero is required, and if measurement is out of the measurement range. The “Query if the Sensor is Being Calibrated” will be described hereinafter.

The first bit in the STATus:OPERation:CALibrating register set represents the calibration status of the sensor.

Set occurrence of the calibration operation event as follows: Event detection in case of jump to calibrated (State 1) from uncalibrated (Status 0), set the positive jump filter to 1.

STAT:OPER:CAL:PTR 2 (Configure the calibration event detection of the sensor as 0 - 1)

Report the calibration event report to the upper level operation status register. Namely, the calibration status bit (0th status) of the STATus:OPERation register set.

STAT:OPER:CAL:ENAB 2 (If the event of the sensor shall be reported, the parameter can be 4 or 6).

Similarly, if the operation status shall be further reported to the 7th bit of the “Status Word”, it is necessary to set to operate the corresponding bit (bit 0) of the positive jump filter and the enable register.

STAT:OPER:CAL:PTR 1 (Configure the calibration event detection as 0 - 1)

STAT:OPER:ENAB 1 (If the lower limit detection event shall be reported, the parameter can be 2048 or 2049).

In this case, the configuration is completed. Next, query if the sensor is being calibrated.

Method 1: Query and calibrate the Bit 1 of the condition register. If non-zero is returned, it means that it is calibrating.

STAT:OPER:CAL:COND?

Method 2: Query the bit 0 of the condition register. If non-zero is returned, it means that it is calibrating.

STAT:OPER:COND?

Method 3: Query the bit 7 of the status word. If non-zero is returned, it means that it is calibrating.

*STB?

2.13 Save & Recall

In order to reduce repeated settings, the S87230 can save up to 5 configuration data to non-volatile memory. The error list, FDO table, zero information, etc. are not saved in this configuration. Except that the error list will not be saved, others will be saved in a hard configuration file and will not be changed along with the user recall.

The relevant SCPIs are:

*SAV <NRf>

*RCL <NRf>

The range of <NRf> is 1 - 5.

Note

Use of commands:

Unless otherwise specified, the commands can be used to set or query.

If a command is only used for setting or query or it is only used to start an event, it will be separately described in the instructions of the command.

Chapter 3 SCPI

3.1 Command Description

This chapter provides detailed command reference information to facilitate remote control, including:

- Complete syntax format and parameter list;
- For the SCPI non-standard commands, list the syntax diagram;
- Detailed function description and instructions of associated command;
- Supported command formats (setting or query);
- Parameter description, including: data type, value range and default value (unit);
- Key path;
- The model of similar instruments compatible to the command. If not indicated, it means that the current command is only applicable to S87230;
- Other descriptions.

The command order items are listed in the common command and instrument subsystem command sections to facilitate the user's query and use.

The description of relevant command suffix in case of remote control in this manual is as shown in the following table:

Table 3.1 Description of the command suffix

Suffix	Value Range	Description
<ch>	1	Channel
<m>	1	Measurement
<w>	1	Window
<t>	1	FDO table

3.2 Common Command (IEEE488.2 Command)

The common command is used to control general functions including the instrument status register, status report, synchronization and data storage. The application method and function of the common command are applicable to different instruments. All the common commands can be identified by the first "*" in the command word, which are defined in details in IEEE488.2.

The explanations and descriptions of the IEEE488.2 common commands are as follows.

- * CLS
- * ESE
- * ESR ?
- * IDN ?
- * OPC
- * RCL
- * RST
- * SAV

- * SRE
- * STB?
- * TRG
- * TST?
- * WAI

Note

Use of commands:

Unless otherwise specified, the commands can be used to set or query.

If a command is only used for setting or query or it is only used to start an event, it will be separately described in the instructions of the command.

***CLS**

Function: Clear the instrument status data structure, including the SCPI register (such as query status and operation status), standard event register, status word and error/event queue.

Query: Not supported

Setting: *CLS

Example: *CLS Clear the instrument status

Error information: None

Reset status: None

***ESE**

Function: Query or set the standard event status enable register. 0 Disable, 1 Enable.

Query: *ESE?

Setting: *ESE <NRf>

The NRf represents the value, a multiple of 2. The bit mapping is as shown in Table 3.2.

Example: *ESE? Query current setting of the register. The returned format is <NR1>, 0 - 255

*ESE 60 Enable 4+8+16+32 bits

Table 3.2 Standard Event Bit Mapping

Bit	Value	Description
0	1	Operation completed
1	2	Not used
2	4	Query error
3	8	Equipment-related error
4	16	Execution error
5	32	Command error
6	64	Not used
7	128	Not used

***ESR?**

Function:	Query the value of the standard event status register, and reset it. See Table 3.2	
Query:	*ESR?	
Setting:	Not supported	
Example:	*ESR?	Query the value of the standard event status register, and reset it.
Error information:	None	
Reset status:	None	

***IDN?**

Function:	Query identification string of S87230.	
Query:	*IDN?	
Setting:	Not supported	
Example:	*IDN?	
Error information:	None	
Reset status:	None	

***OPC**

Function:	When all pending operations are completed, set the operation end bit in the standard event status register.	
Query:	*OPC?	
Setting:	*OPC	
Example:	*OPC?	If the pending operation is completed, return 1; otherwise, wait.
Error information:	None	
Reset status:	None	

***RCL**

Function:	Recall the instrument status in the specified save recall register.
Query:	Not supported
Setting:	*RCL <NRf> The range is 1 - 5
Example:	*RCL 3
Error information:	If the register is not between 1 - 5, it will prompt "-222, Data out of Range".
Reset status:	None

***RST**

Function: Reset the instrument. Please refer to SYSTem:PRESet.

Query: Not supported

Setting: *RST

***SAV**

Function: Save the instrument status in the specified register.

Query: Not supported

Setting: *SAV <NRf>
The range is 1 - 5

Example: *SAV 3

Error information: If the register is not between 1 - 5, it will prompt "-222, Data out of Range".

Reset status: None

***SRE**

Function: Query or set the service request register. 0 Disable, 1 Enable.

Query: *SRE?

Setting: *SRE <NRf>
The NRf represents the value, a multiple of 2. The bit mapping is as shown in Table 3.3.

Example: *SRE? Query current setting of the register. The returned format is <NR1>, 0 - 255

*SRE 316 Respectively set the 2^{nd}, 3^{rd}, 4^{th}, 5^{th} and 8^{th} bits (4+8+16+32+256)

Table 3.3 Mapping of the Service Request Register Bit

Bit	Value	Description
0	1	Not used
1	2	Not used
2	4	Equipment-related
3	8	Questionable status
4	16	Information receiving
5	32	Event status bit
6	64	It must be 0
7	128	Operation status

***STB?**

Function: Query the status word.

Query: *STB?

Setting: Not supported

Example: *STB?

Error information: None

Reset status: None

The bit mapping is as follows:

Table 3.4 Status word

Bit	Value	Description
0	1	Not used
1	2	Equipment-related
2	4	Error/event queue
3	8	Questionable status
4	16	Information receiving
5	32	Event status bit
6	64	Service request
7	128	Operation status

***TRG**

Function: Trigger all channels to be triggered.

Query: Not supported

Setting: *TRG

Example: *TRG

Error information: If the trigger source is not BUS, it will prompt "-211, "Trigger Ignored"".

Reset status: None

***TST?**

Function: Execute self-test, and its time is longer.

Query: *TST?

Setting: Not supported

Example: *TST? If 0 is returned, it means pass; if 1 is returned, it means failure.

Error information: None

Reset status: None

***WAI**

Function: Keep the instrument in a waiting status until one of the followings is satisfied: All pending operations have been completed
Receive the equipment clearing command
Restart.

Query: Not supported

Setting: *WAI

Example: *WAI

Error information: None

Reset status: None

3.3 Instrument Subsystem Command

This section describes S87230 subsystem command in details.

- Calculate (CALCulate)
- Calibration (CALibration)
- Measurement (CONFigure/FETCh/READ/MEASure)
- Format (FORMat)
- Memory (MEMory/MMEMory)
- Sense (SENSe)
- Status (STATus)
- System (SYSTem)
- Initiate/Trigger (INITiate/TRIGger)
- Unit (UNIT)
- Service (SERVice)

3.3.1 Calculation(CALCulate)

The calculate (CALCulate) subsystem is used for post level data processing, and the display offset and relative measurements will be calculated when the switch is turned on.

The command includes:

- :CALCula te[1]:LIMit:CLEar:AUTO
- :CALCulate[1]:LIMit:CLEar[:IMMediate]
- :CALCulate[1]:LIMit:FAIL?
- :CALCulate[1]:LIMit:FCOunt?
- :CALCulate[1]:LIMit:LOWer[:DATA]
- :CALCulate[1]:LIMit:STATe
- :CALCulate[1]:LIMit:UPPer[:DATA]
- :CALCulate[1]:MATH[:EXPRession]
- :CALCulate[1]:MATH[:EXPRession]:CATalogue?
- :CALCulate[1]:RELative[:MAGNitude]:AUTO
- :CALCulate[1]:RELative[:MAGNitude]:VALue?
- :CALCulate[1]:RELative:STATe

:CALCulate[1]:LIMit:CLEar:AUTO

Function: Control the time to reset the limit FCO (failure count).

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

For the ONCE status, if the measurement is not started, 1 will be returned; otherwise, 0 will be returned

For the OFF status, 0 will be returned always

For the ON status, if the measurement is started, 1 will be returned; otherwise, 0 will be returned

Setting: :CALCulate[1]:LIMit:CLEar:AUTO <Boolean Data> | ONCE | 2

The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

For "ON", when following operations are carried out, FCO will be set to 0:

Initialize it with INITiate[:IMMediate] command; Initialize it with INITiate:CONTinuous ON command; Measure it with MEASure? command;

Read the measurement with READ? command. For "OFF", FCO will not be reset.

Example:

CALC:LIM:CLE:AUTO?	Query FCO reset status
CALC:LIM:CLE:AUTO ONCE	Set the reset of FCO during the first initialization.

Reset status: Set it to "ON".

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

Function: Reset FCO (failure count), and the FCO can be obtained by CALCulate[1]:LIMit:FCOunt? query.

Query: Not supported

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

Example: :CALC:LIM:CLE Reset failure count.

:CALCulate[1]:LIMit:FAIL?

Function: Query if it is out of the limit, 1 represents yes and 0 represents no.

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

Setting: Not supported

Example: :CALC:LIM:FAIL? Query the detection status.

:CALCulate[1]:LIMit:FCOunt?

Function: Query the limit detection failure count (FCO). Reset FCO in the following conditions:

1) Reset
2) CALCulate[1]:LIMit:CLEar:IMMediate
3) CALCulate[1]:LIMit:CLEar:AUTO ON

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

Setting: Not supported

Example: :CALC:LIM:FCO? Query the detection failure count.

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

Function: Query or set the lower measurement limit.

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

Setting: :CALCulate[1]:LIMit:LOWer[:DATA] <Numeric Data>

The form of <Numeric Data> includes: DEF, MIN, MAX, NRf, and DEF is used for setting only. If the currently set lower limit value is larger than or equal to the upper limit value, the upper limit value will be automatically adjusted according to the unit, as shown in Table 3.7.

Example: CALC:LIM:LOW? Query the lower limit.

CALC:LIM:LOW 0.2 According to the display unit, set the lower limit to:

0.2 dBm in case of dBm

200 mW in case of W

0.2 dB in case of dB

0.2% in case of %

Reset status: Set to -90 dBm or -90 dB

Table 3.5 Measurement Unit

Measurement Mode	Measurement Type	CALC:REL:STAT OFF		CALC:REL:STAT ON	
		Linear	Logarithmic	Linear	Logarithmic
Single channel	Average, peak	Watt	dBm	%	dB
	PAR	%	dB	%	dB
Ratio	Average, peak, peak-to-average ratio	%	dB	%	dB
Difference	Average, peak	Watt	dBm	%	dB
	PAR	%	dB	%	dB

Table 3.6 Limit Range

	Watt	dBm	%	dB
DEF	1pW	-90	100p%	-120
MIN	1aW	-150	100a%	-180
MAX	1XW	200	100X%	180

Table 3.7 Limit Range Measurement

Lower limit value	Upper limit value			
	Watt	dBm	%	dB
nLow	$nLow \times 10^{0.001}$	$nLow + 0.01$	$nLow \times 10^{0.001}$	$nLow + 0.01$

:CALCulate[1]:LIMit:STATe

Function: Query or set the measurement limit detection switch.

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

Setting: :CALCulate[1]:LIMit:STATe <Boolean Data>

The effective form of <Boolean Data> includes 0, OFF, 1 and ON

Example:

CALC:LIM:STAT?	Query the limit detection switch status.
CALC:LIM:STAT ON	Turn on the limit detection switch.
CALC:LIM:STAT 0	Disable the limit detection.

Limit:

Error information:

Reset status: Disable the limit detection.

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

Function: Query or set the upper measurement limit. Refer to CALCulate[1]:LIMit:LOWer[:DATA]

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

Setting: :CALCulate[1]:LIMit:UPPer[:DATA] <Numeric Data>

The form of <Numeric Data> includes: DEF, MIN, MAX, NRf, and DEF is used for setting only. If the currently set upper limit value is smaller than or equal to the lower limit value, the lower limit value will be automatically adjusted according to the unit, as shown in Table 3.9.

Example:

CALC:LIM:UPP?	Query the upper limit.
CALC:LIM:UPP 8	According to the display unit, set the upper limit to: 8 dBm in case of dBm 8W in case of W 8 dB in case of dB 8% in case of %

Reset status: Set to 90 dBm or 90 dB

Table 3.8 Limit Range

	Watt	dBm	%	dB
DEF	1MW	90	100M%	60
MIN	1aW	-150	100a%	-180
MAX	1XW	200	100X%	180

Table 3.9 Limit Range Measurement

Lower limit value	Upper limit value			
	Watt	dBm	%	dB
nUpp	$nUpp / 10^{0.001}$	*nUpp* - 0.01	$nUpp /10^{0.001}$	*nUpp* - 0.01

:CALCulate[1]:MATH[:EXPRession]

Function:	Query or set the specified measurement expression: Single channel	
Query:	:CALCulate[1]:MATH[:EXPRession]?	
Setting:	:CALCulate[1]:MATH[:EXPRession] <String> The form of the string is as follows: "(SENS1)"	
Example:	CALC1:MATH?	Query the measurement expression.
	CALC:MATH "(SENS1)"	Set the expression to the single channel absolute power measurement.
Reset status:	None	

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

Function:	List all measurement expressions, and separate them by commas. The string is: "(SENS1)"	
Query:	:CALCulate[1]:MATH[:EXPRession]:CATal	
Setting:	Not supported	
Example:	CALC:MATH:CAT?	List all the defined mathematical expressions.

:CALCulate[1]:RELative[:MAGNitude]:AUTO

Function:	Set the reference value for relative measurement. In the CALCulate block, the relative value will be used for the measurement signal after any mathematical calculation and display offset calculation is completed. This value shall be set to ONCE to set the reference value for relative measurement. After the reference value is set, the instruction will return OFF. Set the instruction to ONCE, and change the instruction CALCulate[1]:RELative:STATe to ON. 0|OFF No operation will be carried out. 1|ON Invalid, instrument return error: Invalid parameter. 2|ONCE, valid parameter, and the meaning is
Query:	:CALCulate[1]:RELative[:MAGNitude]:AUTO?
Setting:	:CALCulate[1]:RELative[:MAGNitude]:AUTO ONCE
Example:	CALC:REL:AUTO? 0 will be returned always. CALC:REL:AUTO ONCE Set the reference value for relative measurement.
Error information:	If the parameter is set to 1 or ON, it will prompt “-224, "Invalid Parameter Value"”.

:CALCulate[1]:RELative[:MAGNitude]:VALue?

Function:	Query the reference value for relative measurement.
Query:	:CALCulate[1]:RELative[:MAGNitude]:VALue?
Setting:	Not supported
Example:	CALC:REL:VAL? Query the reference value for relative measurement.

:CALCulate[1]:RELative:STATe

Function:	Query or set the relative measurement switch status. When it is turned on, the relative measurement value set by :CALCulate[1]:RELative[:MAGNitude]:AUTO will be applied for measurement.	
Query:	:CALCulate[1]:RELative:STATe?	
Setting:	:CALCulate[1]:RELative:STATe <Boolean Data>	
Example:	CALC:REL:STAT?	When the relative measurement is enabled, return 1, otherwise, return 0.
	CALC:REL:STAT ON	Enable the relative measurement status.
Reset status:	Disable relative measurement.	

3.3.2 Calibration (CALibration)

The CALibration subsystem is used to control the automatic zero offset setting of the RF power sensor and channel. If no RF signal is loaded onto the sensor, zero setting can be carried out at any time.

The command includes:

- :CALibration[1]:ZERO:AUTO
- :CALibration[1]:ZERO:TYPE

:CALibration[1]:ZERO:AUTO

Function:	Zero the sensor.	
Query:	Not supported	
Setting:	:CALibration[1]:ZERO:AUTO ONCE	
Example:	CAL:ZERO:AUTO ONCE	Zero the sensor.
Error information:	In case of zero error, it will prompt “-231,” Data Questionable; ZERO ERROR"”.	

:CALibration[1]:ZERO:TYPE

Function:	Configure the zero type of S87230.	
Query:	:CALibration[1]:ZERO:TYPE?	
Setting:	:CALibration[1]:ZERO:TYPE <Character Data> The value of <Character Data> is as follows: EXTernal or EXT or 0 represents external zero. INTernal or INT or 1 represents internal zero.	
Example:	CAL:ZERO:TYPE INT	Set the zero type of the sensor to internal.
	CAL:ZERO:TYPE?	Query the zero type of the sensor.
Reset status:	No impact (namely keeping the status before reset, same as below)	

3.3.3 Measurement (CONFigure/FETCh/READ/MEASure)

Generally, if not specified, the linear power unit will be W, the logarithmic power unit will be dBm, the linear power ratio unit will be %, the logarithmic power ratio unit will be dB, and the time unit will be s.

If the returned value is invalid, NAN will be returned (9.91E37). The definition is as shown in IEEE 754.

The difference between the FETCh command and the MEASure command is as follows:

The FETCh command is to query the currently measured value. The execution speed of the FETCh command is irrelated to the power size, and it shall be returned immediately generally. The FETCh command may return intermediate results during the power measurement, rather than the actual power value.

The MEASure command is to start one measurement, and return the measurement value after the measurement is completed. Generally, the lower the power, the longer time the MEASure command will need.

The command includes:

- :CONFigure[1][:SCALar][:POWer][:AC]
- :CONFigure[1][:SCALar][:POWer][:AC]:RELative
- :FETCh[1][:SCALar][:POWer][:AC]?
- :FETCh[1][:SCALar][:POWer][:AC]:RELative?
- :MEASure[1][:SCALar][:POWer][:AC]?
- :MEASure[1][:SCALar][:POWer][:AC]:RELative?
- :READ[1][:SCALar][:POWer][:AC]?
- :READ[1][:SCALar][:POWer][:AC]:RELative?

:CONFigure[1][:SCALar][:POWer][:AC]

Function:	Query or set the power measurement mode.
Query:	:CONFigure[1]?
Setting:	:CONFigure[1][:SCALar][:POWer][:AC] [<Expected Value>[, <Resolution>[, <Source Channel List>]]] Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4; the form of the channel list is: (@1), indicating channel A.
Example:	CONF? Query the power measurement configuration. CONF1 DEF, 3, (@1) Set it to the absolute power measurement, and set the resolution to 3 and source channel to A.
Error information	If no sensor is connected, it will prompt “-241,"Hardware Missing"”.
Reset status:	Set it to measurement of the absolute power, and set the resolution to 3.

:CONFigure[1][:SCALar][:POWer][:AC]:RELative

Function:	Set the absolute power measurement mode, and enable relative measurement.
Query:	Not supported

Setting: :CONFigure[1][:SCALar][:POWer][:AC]:RELative [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and enable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4; the form of the channel list is: (@1), indicating channel A.

Example: CONF:REL DEF, 3, (@2) Set it to the absolute power measurement, and set the resolution to 3 and source channel to B, and enable relative measurement.

Error information:

:FETCh[1][:SCALar][:POWer][:AC]?

Function: Set the absolute power measurement, disable relative measurement, and return the currently measured value. The measurement unit is specified by UNIT[1]:POWer. This command will be returned even though one measurement is not completed. To obtain accurate measurement value, please use meas command.

Query: :FETCh[1][:SCALar][:POWer][:AC]? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4; the form of the channel list is: (@1), indicating channel A.

Setting: Not supported

Example: FETC? Query the absolute power measurement value.

Error information:

:FETCh[1][:SCALar][:POWer][:AC]?

Function: Set the absolute power measurement, enable relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]:POWer:RATio. This command will be returned even though one measurement is not completed. To obtain accurate measurement value, please use MEAS command.

Query: :FETCh[1][:SCALar][:POWer][:AC]:RELative? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4; the form of the channel list is: (@1), indicating channel A.

Setting: Not supported

Example: FETC:REL? Query the relative power measurement value.

Error information:

:MEASure[1][:SCALar][:POWer][:AC]?

Function: Set the absolute power measurement, disable relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]:POWer. Start one measurement, and return the measurement value after completion. The corresponding FETCh command is to return the currently measured value, rather than returning it after the measurement is completed. Please use MEASure command when establishing the test system.

Query: :MEASure[1][:SCALar][:POWer][:AC]? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4;

Setting: Not supported

Example: MEAS? Start one measurement, and return the absolute power measurement value after the measurement is completed.

Error information:

:MEASure[1][:SCALar][:POWer][:AC]:RELative?

Function: Set the absolute power measurement, enable relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]:POWer. Start one measurement, and return the measurement value after completion. The corresponding FETCh command is to return the currently measured value, rather than returning it after the measurement is completed. Please use MEASure command when establishing the test system.

Query: :MEASure[1][:SCALar][:POWer][:AC]:RELative? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4;

Setting: Not supported

Example: MEAS:REL? Start one measurement, and return the relative power measurement value after the measurement is completed.

Error information:

:READ[1][:SCALar][:POWer][:AC]?

Function: Set the absolute power measurement, disable relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]:POWer.

Query: :READ[1][:SCALar][:POWer][:AC]? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4;

Setting: Not supported

Example: READ? Query the absolute power measurement value.

Error information:

:READ[1][:SCALar][:POWer][:AC]:RELative?

Function: Set the absolute power measurement, enable relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]:POWer:RATio.

Query: :READ[1][:SCALar][:POWer][:AC]:RELative? [<Expected Value>[, <Resolution>[, <Source Channel List>]]]

Set it to the absolute power measurement, and disable relative measurement. The expected value and resolution can be expressed in DEF, indicating no change. The range of the resolution is 1 - 4;

Setting: Not supported

Example: READ:REL? Query the relative power measurement value.

Error information:

3.3.4 Format (FORMat)

- :FORMat[:READings]:BORDer
- :FORMat[:READings][:DATA]

:FORMat[:READings]:BORDer

Function: Query or set the transmission order of binary data: Normal or byte exchange. It will be valid only when FORMat[:READings][:DATA] is set to REAL.

Query: :FORMat[:READings]:BORDer?

Setting: :FORMat[:READings]:BORDer <Character Data>

The valid value of <Character Data> is: 0 or NORMal, 1 or SWAPped.

Example: FORM:BORD? Query the transmission order.

FORM:BORD SWAP Set the transmission order to byte exchange.

Reset status: It is set to NORMal

:FORMat[:READings][:DATA]

Function: Query or set the data transmission format: ASCii and REAL (real number)

Query: :FORMat[:READings][:DATA]?

Setting: :FORMat[:READings][:DATA] <Character Data>

The valid value of <Character Data> is: 0 or ASCii, 1 or REAL.

ASCii: The transmission format of the numeric data is <NRf> form.

REAL: The transmission format of the numeric data is IEEE 754 64, and each data has 8 bytes.

Example: FORM? Query the data transmission format.

FORM REAL Set the data transmission format to REAL.

Reset status: Set it to ASCii

3.3.5 Memory (MEMory/MMEMory)

- :MEMory:CATalog[:ALL]?
- :MEMory:CATalog:STATe?
- :MEMory:CATalog:TABLe?
- :MEMory:CLEar[:NAME]
- :MEMory:CLEar:TABLe[1]
- :MEMory:FREE[:ALL]?
- :MEMory:FREE:STATe?
- :MEMory:FREE:TABLe?
- :MEMory:NSTates?
- :MEMory:STATe:CATalog?
- :MEMory:TABLe[1]:DEFine
- :MEMory:TABLe[1]:FREQuency
- :MEMory:TABLe[1]:FREQuency:POINts?
- :MEMory:TABLe[1]:GAIN[:MAGNitude]
- :MEMory:TABLe[1]:GAIN[:MAGNitude]:POINts?
- :MEMory:TABLe:MOVE
- :MEMory:TABLe:SELect

Function:	List the user configuration in the instrument, including save / recall configuration and FDO table. The data format is: <Value 1>, <Value 2>{, <String>} The <Value 1> represents the length of bytes used by the user in the configuration, and <Value 2> represents the length of unused bytes. The form of each <String> is as follows: <String i>, <Type>, <Length> The <String i> indicates name of the user configuration. The <Type> indicates type of the user configuration, TABL represents FDO table, and STAT represents save recall configuration. The <Length> indicates the byte length of the configuration item.	
Query:	:MEMory:CATalog[:ALL]?	
Setting:	Not supported	
Example:	MEM:CAT?	List all the user configurations in the instrument.
Reset status:	None	

:MEMory:CATalog:STATe?

Function: List the save recall configuration in the instrument. The data format is: <Value 1>, <Value 2>{, <String>}
The <Value 1> represents the length of bytes used by the user in the configuration, and <Value 2> represents the length of unused bytes. The form of each <String> is as follows:
<String i>, <Type>, <Length>
The <String i> indicates name of the user configuration.
The <Type> indicates type of the user configuration, and STAT represents the save recall configuration.
The <Length> indicates the byte length of the configuration item.

Query: :MEMory:CATalog:STATe?

Setting: Not supported

Example: MEM:CAT:STAT? List the save recall configuration in the instrument.

:MEMory:CATalog:TABLe?

Function: List FDO table in the instrument.
The data format is: <Value 1>, <Value 2>{, <String>}
The <Value 1> represents the length of bytes used by the user in the configuration, and <Value 2> represents the length of unused bytes. The form of each <String> is as follows:
<String i>, <Type>, <Length>
The <String i> indicates name of the user configuration.
The <Type> indicates type of the user configuration, and TABL represents FDO table. The <Length> indicates the byte length of the configuration item.

Query: :MEMory:CATalog:TABLe?

Setting: Not supported

Example: MEM:CAT:TABL? List FDO table in the instrument.

:MEMory:CLEar[:NAME]

Function: It is used to reset FDO table and save recall table specified in the instrument.

Query: Not supported

Setting: :MEMory:CLEar[:NAME] <String>
The <String> represents name of FDO table or save recall table.

Example: MEM:CLE "fdo0" Clear fdo0 FDO table.

Description:
1) It will be automatically used if FDO table is enabled.
2) The English name of the save recall status is “State1”, “State2”……“State5”, and the name can’t be changed.
3) The FDO table will be cleared if the specified name can be found in the FDO list;
4) The status table will be cleared if the specified name can be found in the save recall list.
5) If “State1” FDO exists, recall MEM:CLE “State1” to clear FDO table and status table.

6) In the remote control setting, the status name of the save / recall is irrelated to the language switch of the interface, namely, the following two commands will be used to clear the State 1 (English name is recommended):

MEM:CLE "State1"

Error information: If there is no specified name, it will prompt “-224, "Invalid Parameter Value"”.

Reset status: None

:MEMory:CLEar:TABLe[1]

Function: Clear the specified FDO table.

Query: Not supported

Setting: :MEMory:CLEar:TABLe[1]

Example: MEM:CLE:TABL Clear the FDO table (10 totally).

:MEMory:FREE[:ALL]?

Function: Query the total number of unused bytes and the number of used bytes in the user configuration space.

The form of the returned string: <Number of Unused Bytes>, <Number of Used Bytes>

Query: :MEMory:FREE[:ALL]?

Setting: Not supported

Example: MEM:FREE?

:MEMory:FREE:STATe?

Function: Query the total number of unused bytes and the number of used bytes in the save recall space.

The form of the returned string: <Number of Unused Bytes>, <Number of Used Bytes>

Query: :MEMory:FREE:STATe?

Setting: Not supported

Example: MEM:FREE:STAT?

:MEMory:FREE:TABLe?

Function: Query the total number of unused bytes and the number of used bytes in FDO table space. The form of the returned string: <Number of Unused Bytes>, <Number of Used Bytes>

Query: :MEMory:FREE:TABLe?

Setting: Not supported

Example: MEM:FREE:TABL?

:MEMory:NSTates?

Function: Query number of the save / recall status, and 10 will be returned always.

Query: :MEMory:NSTates?

Setting: Not supported

Example: MEM:NST?

:MEMory:STATe:CATalog?

Function: List name of all the save recall status.
Query: :MEMory:STATe:CATalog?
Setting: Not supported
Example: MEM:STAT:CAT?

:MEMory:TABLe[1]:DEFine

Function: Query or set name of the specified FDO table.
Query: :MEMory:TABLe[1]:DEFine?
Setting: :MEMory:TABLe[1]:DEFine <String>
Example: MEM:TABL:DEF? Query name of the FDO table.
MEM:TABL:DEF "fdo0" Name the FDO table as fdo0.
Error information: If the specified name exists, it will prompt "-257, "File Name Error"".

:MEMory:TABLe[1]:FREQuency

Function: Query or set the frequency list of the specified FDO table.
When the frequency list is set, the previous frequency list will be cleared, and the frequency list must be in ascending order.
The relevant command is :MEMory:TABLe[1]:GAIN[:MAGNitude]
Query: :MEMory:TABLe[1]:FREQuency?
Setting: :MEMory:TABLe[1]:FREQuency <Numeric Data 1>{, <Numeric Data n>}
It is necessary to ensure that the frequency list covers the frequency range of the sensor, and if the frequency of the measured signal is out of the range of the list, the value at the lowest or the highest frequencies will be used.
The number of the maximum frequency point is 80.
Example: MEM:TABL:FREQ? Query the frequency list of the FDO table.
MEM:TABL:FREQ 50MHz, 40GHz Set the FDO table frequency to 50 MHz, 40 GHz
Error information: If the number of points in the frequency list is more than 80, it will prompt "-108, "Parameter Not Allowed"".
If the frequency list is not in ascending order, it will prompt "-220," Parameter Error"".
Reset status: None

:MEMory:TABLe[1]:FREQuency:POINts?

Function: Query number of the frequency point of FDO table.
Query: :MEMory:TABLe[1]:FREQuency:POINts?
Setting: Not supported
Example: MEM:TABL:FREQ:POIN? Query number of the frequency point of FDO table.

:MEMory:TABLe[1]:GAIN[:MAGNitude]

Function: Query or set the amplitude gain list of the specified FDO table. When the gain list is set, the previous gain list will be cleared.
The relevant command is :MEMory:TABLe[1]:FREQuency

Query: :MEMory:TABLe[1]:GAIN[:MAGNitude]

Setting: :MEMory:TABLe[1]:GAIN[:MAGNitude] <Numeric Data 1>{, <Numeric Data n>} The number of the maximum gain point is 80.
The unit is PCT, namely 100 represents 100%
The amplitude gain range is 1.0e-009 - 1.0e+009, namely -90 dB90 dB.

Example: MEM:TABL:GAIN? Query the gain list of the FDO table.
MEM:TABL:GAIN98, 102 Set the gain list of the FDO table to 98%, 102%.

:MEMory:TABLe[1]:GAIN[:MAGNitude]:POINts?

Function: Query number of the amplitude gain point of FDO table.

Query: :MEMory:TABLe[1]:GAIN[:MAGNitude]:POINts?

Setting: Not supported

Example: MEM:TABL:GAIN:POIN? Query number of the amplitude gain point of FDO table.

:MEMory:TABLe:MOVE

Function: Rename the specified FDO table. This command needs to know name of FDO table to be modified in advance.
The relevant command is MEMory:TABLe[1]:DEFine The MEMory:TABLe[1]:DEFine is recommended.

Query: Not supported

Setting: :MEMory:TABLe:MOVE <String 1>, <String 2>
The <String 1> indicates name of FDO table to be modified. The <String 2> indicates name of the modified FDO table.

Example: MEM:TABL:MOVE "fdo0", "fdo1"

Error information: If FDO table specified by the first parameter doesn" t exist, it will prompt “-256, "File Name Not Found"”.
If FDO table specified by the second parameter doesn" t exist, it will prompt “-257, "File Name Error"”.
If the length of the second parameter exceeds 16 characters, it will prompt “-257, "File Name Error"”

:MEMory:TABLe:SELect

Function: Query or set the current FDO table. It is different from AV2434. As the sensor calibration table is supported in AV2434, it can be selected with this command. Only FDO table is supported in this case.

Query: :MEMory:TABLe:SELect?

Setting:	:MEMory:TABLe:SELect <String>	
Example:	MEM:TABL:SEL?	Return name of the current FDO table.
	MEM:TABL:SEL "fdo0"	Set the current FDO table to fdo0.
Error information:	If FDO table specified by the parameter doesn" t exist, it will prompt "-224, "Invalid Parameter Value"".	

3.3.6 Sense (SENSe)

- [:SENSe[1]:AVERage[1]:COUNt
- [:SENSe[1]:AVERage:COUNt:AUTO
- [:SENSe[1]:AVERage:SDETect
- [:SENSe[1]:AVERage[1][:STATe]
- [:SENSe[1]:CORRection:CSET2[:SELect]
- [:SENSe[1]:CORRection:CSET:STATe
- [:SENSe[1]:CORRection:FDOFfset[:INPut][:MAGNitude]?
- [:SENSe[1]:CORRection:GAIN[1]|2|3|4:INPut][:MAGNitude]
- [:SENSe[1]:CORRection:GAIN[1]|2|3|4:INPut]:STATe
- [:SENSe[1]:FREQuency[:CW|FIXed]
- [:SENSe[1]:POWer:AC:RANGe
- [:SENSe[1]:POWer:AC:RANGe:AUTO

[:SENSe[1]:AVERage[1]:COUNt

Function:	Query or set the average number of times of the channel.	
Query:	[:SENSe[1]:AVERage[1]:COUNt?[MIN|MAX]	
Setting:	[:SENSe[1]:AVERage[1]:COUNt <Numeric Data> The valid value of the numeric data includes: DEF, MIN, MAX and <NRf>, and DEF and <NRf> are only used for setting. The range of <NRf> is 1 - 1024, The DEF is 8%, The MIN is 1, The MAX is 1,024.	
Example:	SENS1:AVER:COUN?	Query the average number of times.
	SENS1:AVER:COUN? MAX	Query the settable maximum value of the average number of times.
	SENS:AVER:COUN 28	Set average number of times to 28.
Reset status:	Set the average number of times to 8.	

[:SENSe[1]:AVERage:COUNt:AUTO

Function:	Query or set the automatic average status of the channel. This function is only valid for the continuous wave sensor.
Query:	[:SENSe[1]:AVERage:COUNt:AUTO?
Setting:	[:SENSe[1]:AVERage:COUNt:AUTO <Boolean Data> The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

Example:	SENS1:AVER:COUN:AUTO?	Query the automatic average status.
	SENS:AVER:COUN:AUTO 1	Enable the automatic average.
Limit:	It is only valid for the continuous wave sensor.	
Reset status:	The automatic average of the continuous wave sensor is set to ON.	

[:SENSe[1]:AVERage:SDETect

Functions: Query or set the step detection status of the channel. This function is only valid for the continuous wave sensor.

The last four average measurement values in the automatic average mode are used to compare with the value of the whole filter. When the average difference between these two is large than 15%, clear the digital filter. After that, the filter will start saving new measurement values. The function is intended to shorten the filtering time when the power changes obviously.

Query: [:SENSe[1]:AVERage:SDETect?

Setting: [:SENSe[1]:AVERage:SDETect <Boolean Data>

The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

Example:	SENS1:AVER:SDET?	Query the step detection status.
	SENS:AVER:SDET 1	Enable the step detection.
Limit:	It is only valid for the continuous wave sensor.	
Reset status:	The step detection of the continuous wave sensor is set to ON.	

[:SENSe[1]:AVERage[1][:STATe]

Function: Query or set the average switch status of the channel.

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

Setting: [:SENSe[1]:AVERage[1][:STATe] <Boolean Data>

The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

Example:	SENS1:AVER?	Query the average switch status.
	SENS:AVER 1	Turn on average switch.
Reset status:	Set the average status to ON.	

[:SENSe[1]:CORRection:CSET2[:SELect]

Function: In this case, only the [:SENSe[1]:CORRection:CSET2[:SELect] is supported. Select FDO table.

The data of FDO table is shared by two channels, but the switch status is separate.

Query: [:SENSe[1]:CORRection:CSET2[:SELect]?

Setting: [:SENSe[1]:CORRection:CSET2[:SELect] <String> The <String> indicates name of FDO table.

Example:	SENS:CORR:CSET2?	Query the currently selected FDO table.
	SENS:CORR:CSET2 "fdo0"	Select FDO table with a name of “fdo0”.
Error information	If FDO table specified by the parameter doesn’t exist, it will prompt “-256, "File Name Not Found"”.	

Reset status: No impact.

[:SENSe[1]:CORRection:CSET:STATe

Function: In this case, only the [:SENSe[1]:CORRection:CSET:STATe is supported. Query or set the enable status of FDO table.

The data of FDO table is shared by the channels, but the enable status is separate.

Query: [:SENSe[1]:CORRection:CSET:STATe?

Setting: [:SENSe[1]:CORRection:CSET:STATe <Boolean Data>

The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

Example: SENS1:CORR:CSET:STAT? Query the enable status of the FDO table.

SENS:CORR:CSET:STAT 0 Disable the FDO table.

Error information: If the frequency points and amplitude gain (offset) points in the currently selected FDO table are different, it will prompt "-226," Lists Not Same Length"".

If the point of the currently selected FDO table is 0, it will prompt "-221, "Settings Conflict"".

Reset status: It will not affect the enable status of FDO table.

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

Function: Query the currently used frequency response offset factor. The unit is PCT, namely 100 represents 100%.

Query: [:SENSe[1]:CORRection:FDOFfset[:INPut][:MAGNitude]?

Setting: Not supported

Example: CORR:FDOF? Query the currently used frequency response offset factor.

[:SENSe[1]:CORRection:GAIN[1]|2|3|4:INPut][:MAGNitude]

Function: The equivalent form of GAIN[1] is CFACtor, which represents the calibration factor of the sensor. The former conforms to SCPI specification, while the latter doesn" t conform to it. The GAIN[1] is not supported in this case.

The GAIN2 represents the channel offset. It is supported in this case. Both query and setting are supported.

The equivalent form of GAIN3 is DCYCle, indicating the duty cycle. The GAIN3 is not supported in this case.

Query: [:SENSe[1]:CORRection:GAIN2|4[:INPut][:MAGNitude]? [MIN|MAX]

Setting: [:SENSe[1]:CORRection:GAIN2[:INPut][:MAGNitude] <Numeric Data>

The valid value of the numeric data includes: DEF, MIN, MAX and <NRf>, and DEF and <NRf> are only used for setting.

The range of <NRf> is -100 - 100 dB, The DEF is 0 dB,

The MIN is -100 dB, The MAX is 100 dB.

Example: SENS:CORR:GAIN2? Query the channel offset.

CORR:GAIN2? MAX Query the maximum settable channel offset value.

SENS1:CORR:GAIN4? Query the frequency response offset value.

CORR:GAIN2 3.6 Set the channel offset to 3.6 dB.

[:SENSe[1]:CORRection:GAIN[1]|2|3|4:INPut]:STATe

Functions: The equivalent form of GAIN[1] is CFACtor, which represents the calibration factor of the sensor. The former conforms to SCPI specification, while the latter doesn't conform to it. The GAIN[1] is not supported in this case.
The GAIN2 represents the channel offset.
The equivalent form of GAIN3 is DCYCle, indicating the duty cycle. The GAIN3 is not supported in this case.
The equivalent form of GAIN4 is FDOFfset, indicating the FDO factor. The query and setting of GAIN4 is not supported in this case.

Query: [:SENSe[1]:CORRection:GAIN2[:INPut]:STATe?

Setting: [:SENSe[1]:CORRection:GAIN2[:INPut]:STATe <Boolean Data> The effective form of <Boolean Data> includes 0, OFF, 1 and ON.

Example: SENS:CORR:GAIN2:STAT? Query the channel offset enable status.
CORR:GAIN2:STAT 1 Enable the channel offset.

[:SENSe[1]:FREQuency[:CW|FIXed]

Function: Query or set the frequency.

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

Setting: [:SENSe[1]:FREQuency[:CW|FIXed] <Numeric Data>
The valid value of the numeric data includes: DEF, MIN, MAX and <NRf>, and DEF and <NRf> are only used for setting. The range of <NRf> is 1e3 - 1e12, The DEF is 50 MHz,
The MIN is 1 kHz,
The MAX is 1,000 GHz.

Example: FREQ? Query the frequency.
FREQ? MAX Query the maximum settable frequency.
SENS:FREQ 8GHz Set the frequency to 8 GHz

Reset status: Set it to 50 MHz.

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

Function: Query or set range of the sensor. This function is only valid for the continuous wave sensor. 0 represents low range and 1 represents high range.
When the range is set with this command, the automatic range status will be changed to manual, namely, set [SENSe[1]|SENSe:POWer:AC:RANGe:AUTO to OFF.

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

Setting: [:SENSe[1]:POWer:AC:RANGe <0|1>

Example: SENS:POW:AC:RANG? Query the current range.
POW:AC:RANG 1 Set the range to the high range.

Limit: It is only valid for the continuous wave sensor.

Reset status:	Set the range to high range.

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

Function:	Query or set the automatic range switch status of the sensor. This function is only valid for the continuous wave sensor. 0 represents manual, and 1 represents automatic. When the automatic range is enabled, the instrument will automatically adjust the range according to the power signal, and adopt the optimal range for measurement. Relevant command: [:SENSe[1]:POWer:AC:RANGe.
Query:	[:SENSe[1]:POWer:AC:RANGe:AUTO?
Setting:	[:SENSe[1]:POWer:AC:RANGe:AUTO <Boolean Data> The effective form of <Boolean Data> includes 0, OFF, 1 and ON.
Example:	SENS:POW:AC:RANG:AUTO? Query the automatic range switch. POW:AC:RANG:AUTO 0 Disable the automatic range.
Limit:	It is only valid for the continuous wave sensor.
Reset status:	Set it to automatic range.

3.3.7 Status (STATus)

The status subsystem command will detect the status of the instrument through monitoring the equipment status register, operation status register and questionable register.

- :MEMory:CATalog[:ALL]?
- :STATus:DEVice:CONDition?
- :STATus:DEVice:ENABle
- :STATus:DEVice[:EVENt]?
- :STATus:DEVice:NTRansition
- :STATus:DEVice:PTRansition
- :STATus:OPERation:CALibrating[:SUMMary]:CONDi tion?
- :STATus:OPERation:CALibrating[:SUMMary]:ENABle
- :STATus:OPERation:CALibrating[:SUMMary][:EVENt]?
- :STATus:OPERation:CALibrating[:SUMMary]:NTRansition
- :STATus:OPERation:CALibrating[:SUMMary]:PTRansition
- :STATus:OPERation:CONDition?
- :STATus:OPERation:ENABle
- :STATus:OPERation[:EVENt]?
- :STATus:OPERation:NTRansition
- :STATus:OPERation:PTRansition
- :STATus:OPERation:LLFail[:SUMMary]:CONDition?
- :STATus:OPERation:LLFail[:SUMMary]: ENABle
- :STATus:OPERation:LLFail[:SUMMary][:EVENt]?
- :STATus:OPERation:LLFail[:SUMMary]:NTRansition

- :STATus:OPERation:LLFail[:SUMMary]:PTRansition
- :STATus:OPERation:SENSe[:SUMMary]:CONDition?
- :STATus:OPERation:SENSe[:SUMMary]:ENABle
- :STATus:OPERation:SENSe[:SUMMary][:EVENt]?
- :STATus:OPERation:SENSe[:SUMMary]:NTRansition
- :STATus:OPERation:SENSe[:SUMMary]:PTRansition
- :STATus:OPERation:ULFail[:SUMMary]:CONDition?
- :STATus:OPERation:ULFail[:SUMMary]:EN ABle
- :STATus:OPERation:ULFail[:SUMMary][:EVENt]?
- :STATus:OPERation:ULFail[:SUMMary]:NTRansition
- :STATus:OPERation:ULFail[:SUMMary]:PTRansition
- :STATus:PRESet
- :STATus:QUEStionable:CALibration[:SUMMary]:CONDition?
- :STATus:QUEStionable:CALibration[:SUMMary]:ENABle
- :STATus:QUEStionable:CALibration[:SUMMary][:EVENt]?
- :STATus:QUEStionable:CALibration[:SUMMary]:NTRansition
- :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition
- :STATus:QUEStionable:CONDition?
- :STATus:QUEStionable:ENABle
- :STATus:QUEStionable[:EVENt]?
- :STATus:QUEStionable:NTRansition
- :STATus:QUEStionable:PTRansition
- :STATus:QUEStionable:POWer[:SUMMary]:CONDition?
- :STATus:QUEStionable:POWer[:SUMMary]:ENABle
- :STATus:QUEStionable:POWer[:SUMMary][:EVENt]?
- :STATus:QUEStionable:POWer[:SUMMary]: NTRansition
- :STATus:QUEStionable:POWer[:SUMMary]:PTRansition

Table 3.10 Command or Event Affecting the Status Register

Status register	*RST	*CLS	Power-on	STATus:PRESet
SCPI jump filter (NTR and PTR)	No impact	No impact	Preset	Preset
SCPI enable register	No impact	No impact	Preset	Preset
SCPI event register	No impact	Reset	Reset	No impact
SCPI error/event queue enable	No impact	No impact	Preset	Preset
SCPI error/event queue	No impact	Reset	Reset	No impact
IEEE488.2 register ESE SRE	No impact	No impact	Reset	No impact
IEEE488.2 register SESR STB	No impact	Reset	Reset	No impact

Description of preset status: The preset value of PTR is 0x7fff (32767); NTR and enable registers are reset.

- Command set
- Jump filter
- Summary of equipment status register
- Summary of operation status register
- Summary of calibration operation status register
- Summary of lower limit detection operation status register
- Summary of sense operation status register
- Description of trigger operation status register
- Description of upper limit detection operation status register
- Description of questionable status register
- Description of calibration questionable status register
- Description of power questionable status register

3.3.7.1 Command set

Query or set the content of the status register through the following command set:

:CONDition?

Query the condition register value of the status register, and the format of the returned value is <NR1>. The range is 0 - 32767. The value of the condition register will be kept unchanged after the query.

:ENABle <NRf>|<Non-decimal>

Query or set the event enable register of the status register, and the highest bit (Bit15) is always 0. [:EVENt?]

Query the event register of the status register, and reset it after the query.

:NTRansition <NRf>|<Non-decimal>

Query or set the negative jump filter of the status register, and the highest bit is always 0.

:PTRansition <NRf>|<Non-decimal>.

Query or set the positive jump filter of the status register, and the highest bit is always 0. The status register supported in this case includes:

STATus:DEVice STATus:OPERation

STATus:OPERation:CALibrating[:SUMMary] STATus:OPERation:LLFail[:SUMMary]

STATus:OPERation:SENSe[:SUMMary] STATus:OPERation:ULFail[:SUMMary] STATus:QUEStionable

STATus:QUEStionable

STATus:QUEStionable:CALibration[:SUMMary] STATus:QUEStionable:POWer[:SUMMary]

For example:

The :CONDition? Can be used to query and calibrate the operation register

STATus:OPERation:CALibrating[:SUMMary]

STATus:OPERation:CALibrating[:SUMMary]:CONDition?

The :ENABle can be used to query or set the calibration operation register

STATus:OPERation:CALibrating[:SUMMary]

STATus:OPERation:CALibrating[:SUMMary]:ENABle

3.3.7.2 Jumper filter

For description of the jump filter, please refer to section 9.2 of SCPI-99. The brief introduction is given below.

1） Positive jump (PTR): When the condition is FALSE - TRUE, the event is set to TRUE.

2） Negative jump (NTR): When the condition is TRUE - FALSE, the event is set to TRUE.

3） Positive jump or negative jump: When the condition is FALSE - TRUE or TRUE - FALSE, the event is set to TRUE.

4） The event report will be disabled by resetting the positive and negative jump register.

3.3.7.3 Description of equipment status register (STATus:DEVice)

Table 3.11 Description of Equipment Status Register

Bit	Value	Definition
0	1	Not used
1	2	Sensor detection status
2	4	Reserved
3	8	Error status of the sensor
4	16	Reserved
5	32	Reserved
6	64	Reserved
7-14	-	Not used
15	-	It is always 0

1) Bit 1 and expression of the sensor detection status.

- Returned value of STATus:DEVice:CONDition?: 1 means that sensor is detected and 0 means that no sensor is connected.
- Returned value of STATus:DEVice[:EVENt]?: 1 means that sensor is connected or removed. 0 means no occurrence. The event register will be reset after the query.
- When STATus:DEVice:NTRansition is set to 1, if it is detected that the sensor is removed, set STATus:DEVice[:EVENt] to 1.
- When STATus:DEVice:PTRansition is set to 1, if it is detected that the sensor is connected, set STATus:DEVice[:EVENt] to 1.

2) Bit 3 means sensor error. 1 means error and 0 means that no error is found.

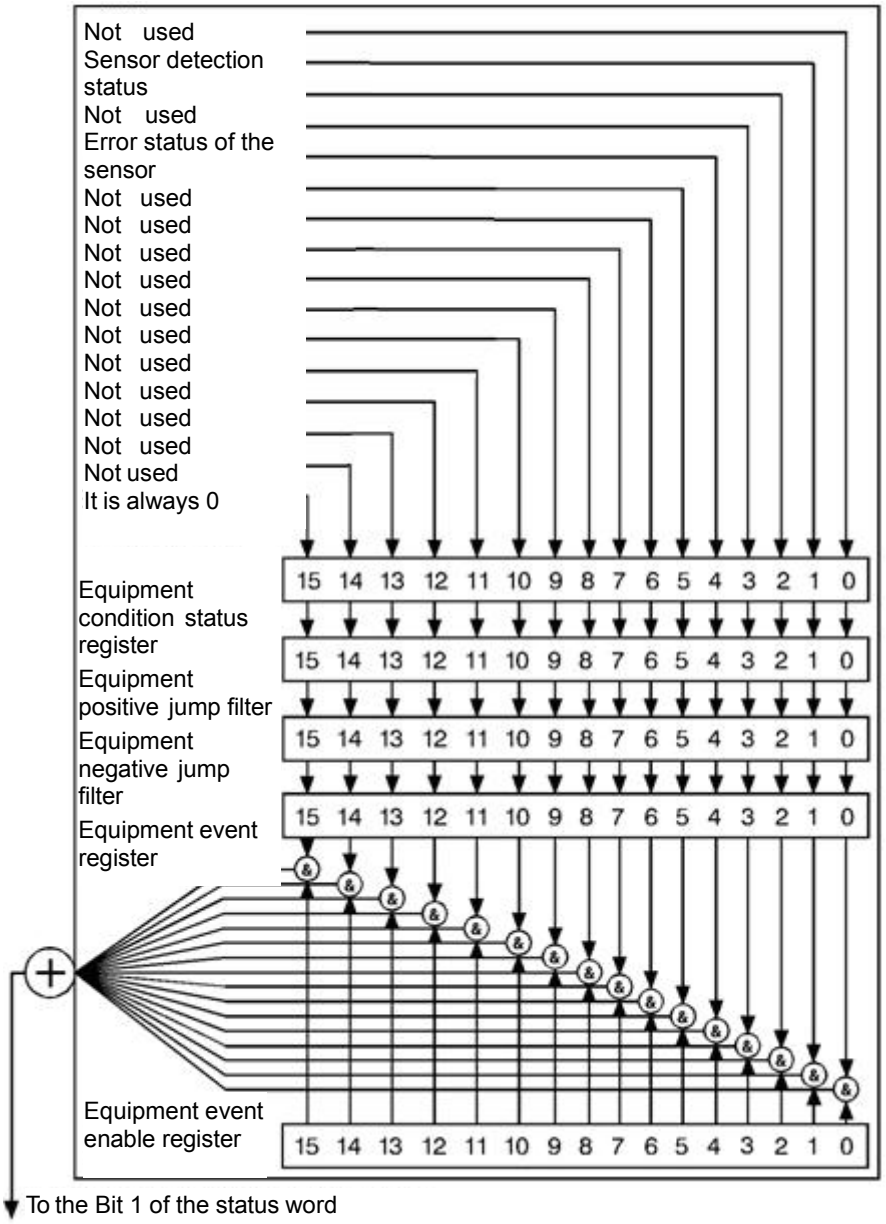


Fig.3.1 Equipment Status Register

3.3.7.4 Description of operation status register (STATus:OPERation)

Table 3.12 Description of Operation Status Register

Bit	Value	Definition
0	1	Calibration summary
1-4	-	Not used
5	32	Summary of waiting for trigger
6-9	-	Not used
10	1024	Summary of sense
11	2048	Summary of lower limit detection failure (LLF)
12	4096	Summary of upper limit detection failure (ULF)
13-14	-	Not used

15	-	It is always 0

It includes 6 groups of operation registers:

- STATus:OPERation
- STATus:OPERation:CALibrating[:SUMMary]
- STATus:OPERation:LLFail[:SUMMary]
- STATus:OPERation:SENSe[:SUMMary]
- STATus:OPERation:TRIGger[:SUMMary]
- STATus:OPERation:ULFail[:SUMMary]

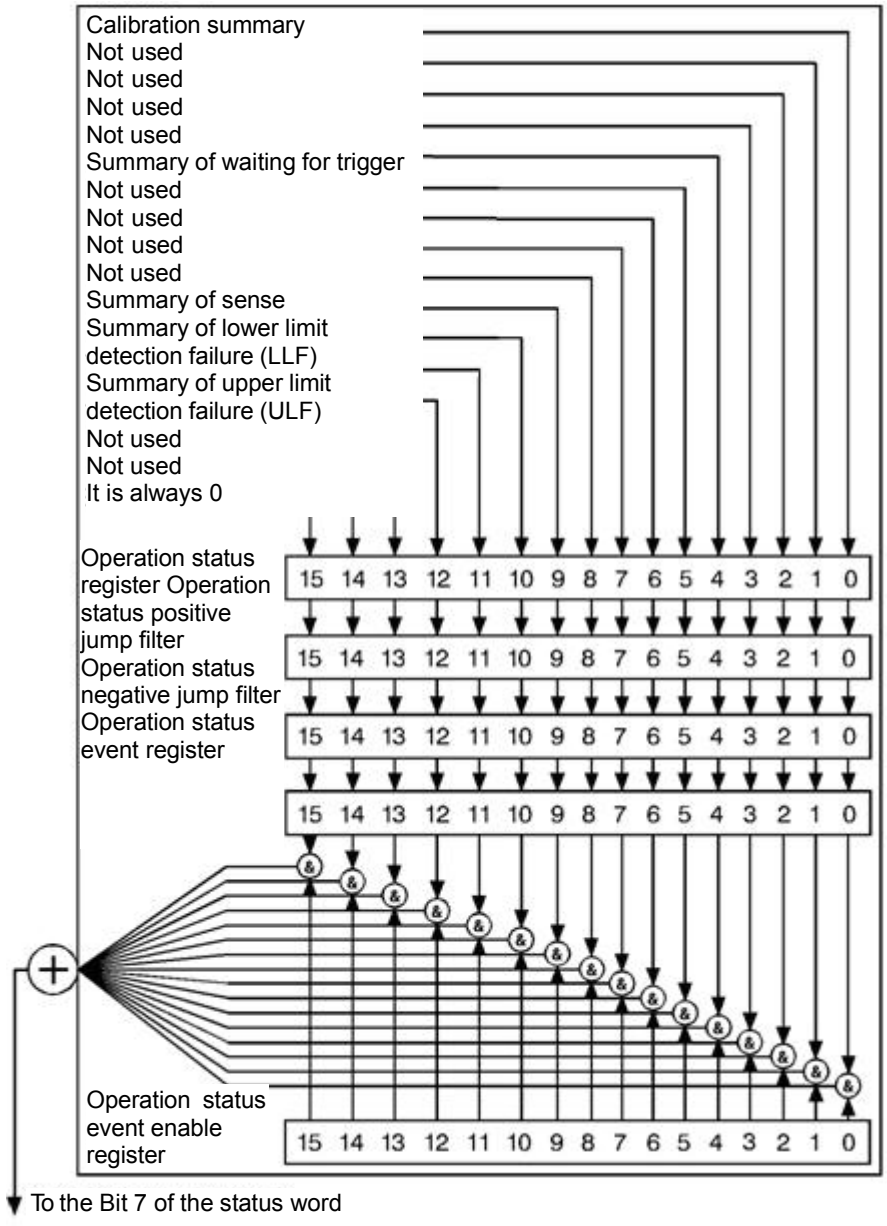


Fig.3.2 Operation Status Register

3.3.7.5 Summary of calibration operation status register (STATus:OPERation:CALibrating[:SUMMary])

Table 3.13 Summary of Calibration Operation Status Register

Bit	Value	Definition

0	1	Not used
1	2	Calibration state
2	4	Reserved
3-14	-	Not used
15	-	It is always 0

Bit 1 represents the calibration status of the channel. 1 means that zero or calibration is being carried out, 0 means that calibration is completed or no zero or calibration is carried out.

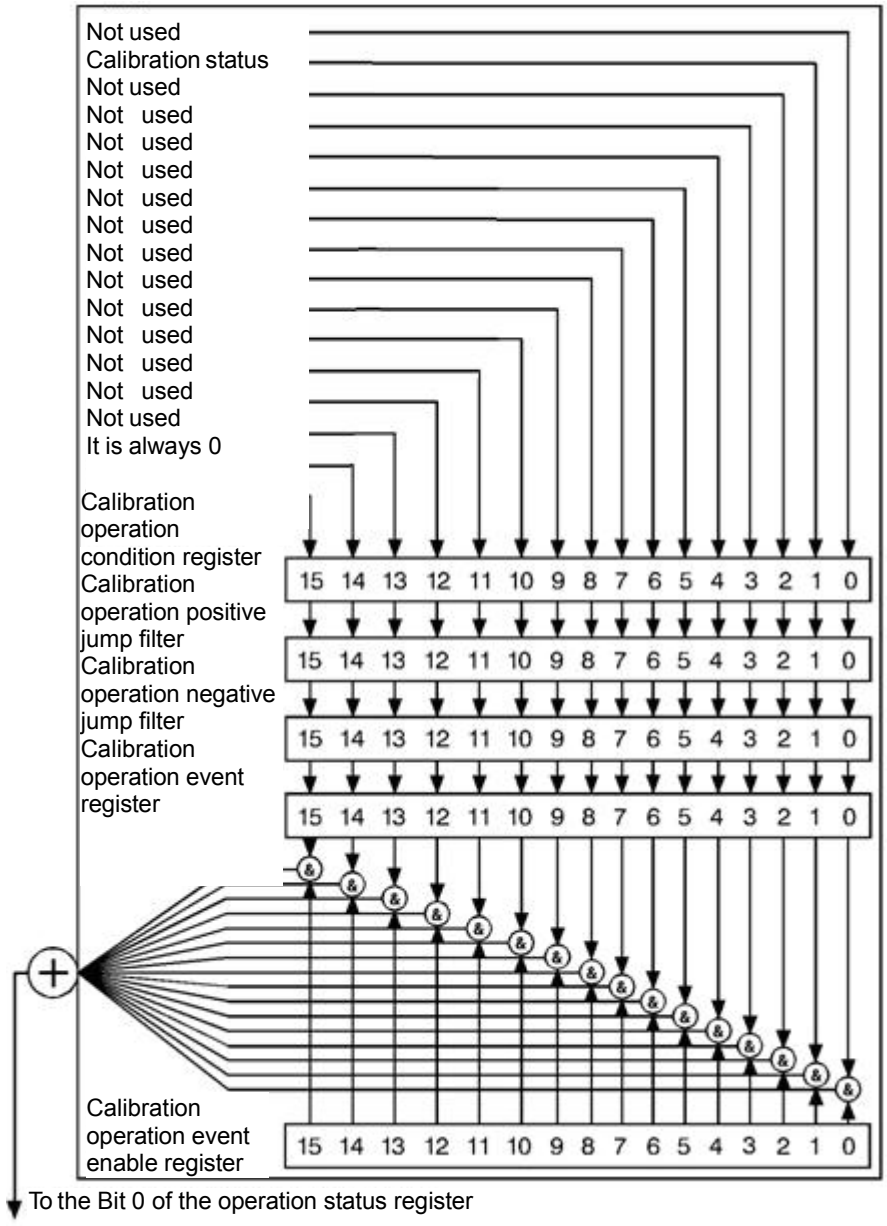


Fig.3.3 Calibration Operation Status Register

3.3.7.6 Summary of lower limit detection operation status register (STATus:OPERation:LLFail[:SUMMary])

Table 3.14 Summary of Lower Limit Detection Operation Status Register

Bit	Value	Definition

0-2	-	Not used
3	8	Lower limit detection status
4	16	Reserved
5	32	Reserved
6	64	Reserved
7-14	-	Not used
15	-	It is always 0

Bit 3 represents the failure detection status of the lower limit, and 1 means that it is out of the lower limit.

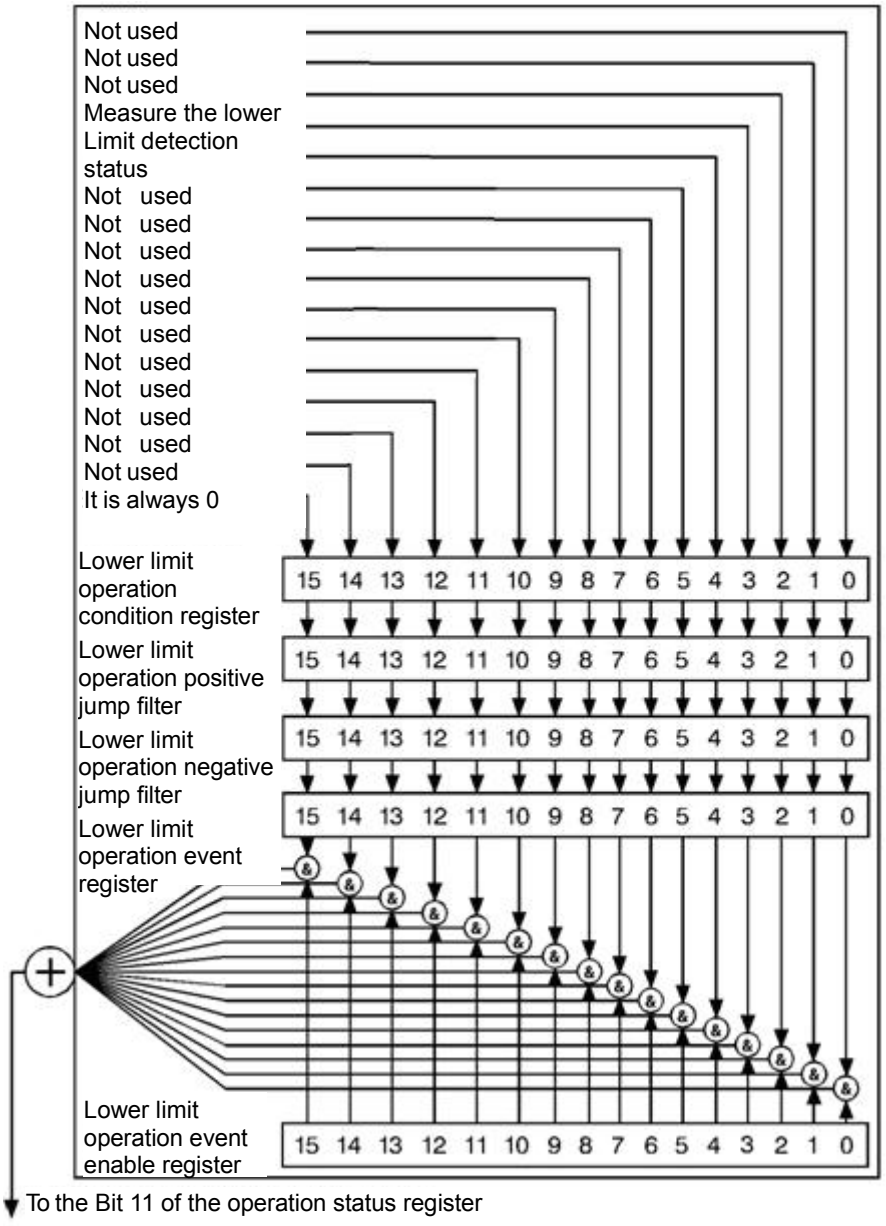


Fig.3.4 Lower Limit Detection Operation Status Register

3.3.7.7 Summary of sense operation status register (STATus:OPERation:SENSe[:SUMMary])

Table 3.15 Summary of Sense Operation Status Register

Bit	Value	Definition
0	1	Not used
1	2	Reading sensor EEPROM status
2	4	Reserved
3-14	-	Not used
15	-	It is always 0

Bit 1 represents the reading sensor EEPROM status. 1 means that the sensor is being read.

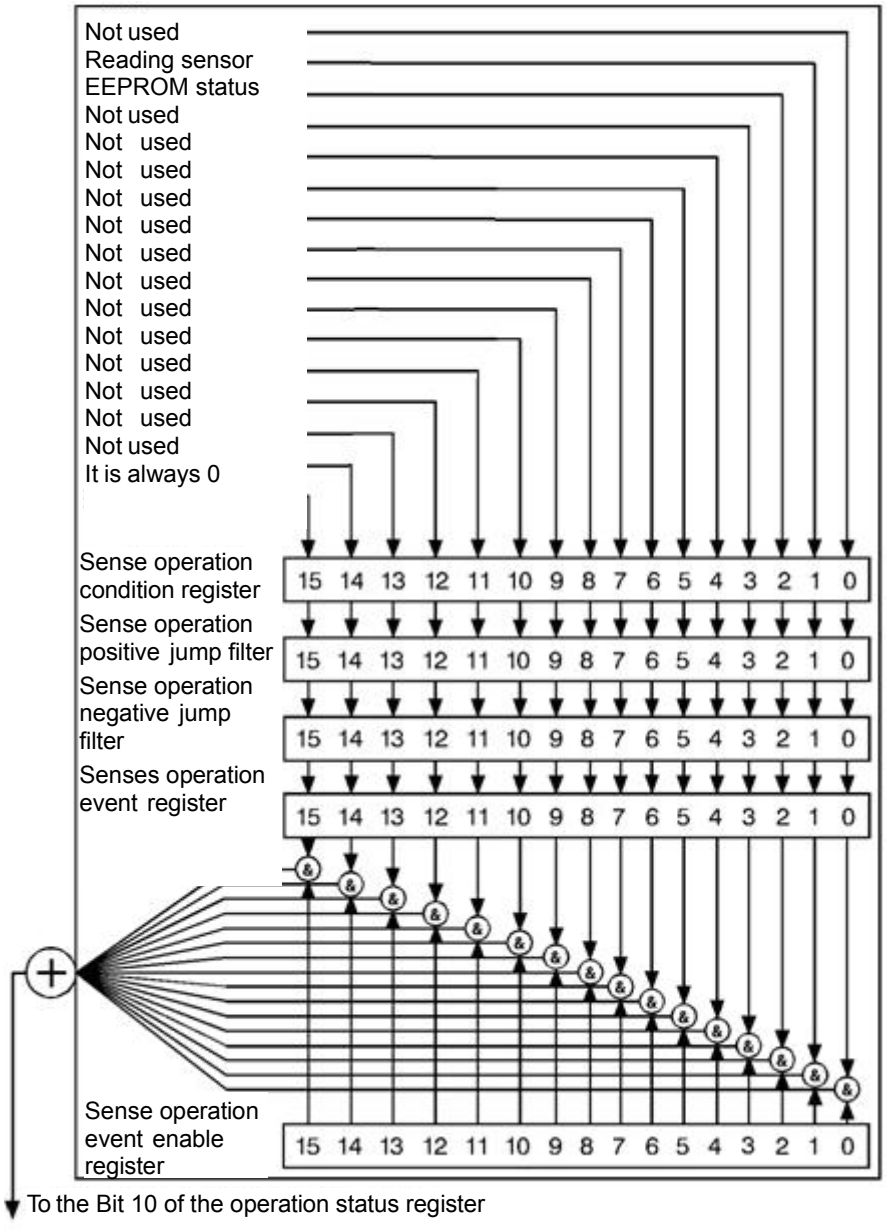


Fig.3.5 Sense Operation Status Register

3.3.7.8 Summary of trigger operation status register (STATus:OPERation:SENSe[:SUMMary])

Table 3.15 Summary of Sense Operation Status Register

Bit	Value	Definition
0	1	Not used
1	2	Trigger status
2	4	Reserved
3-14	-	Not used
15	-	It is always 0

Bit 1 represents the trigger waiting status. 1 means that it is waiting for trigger.

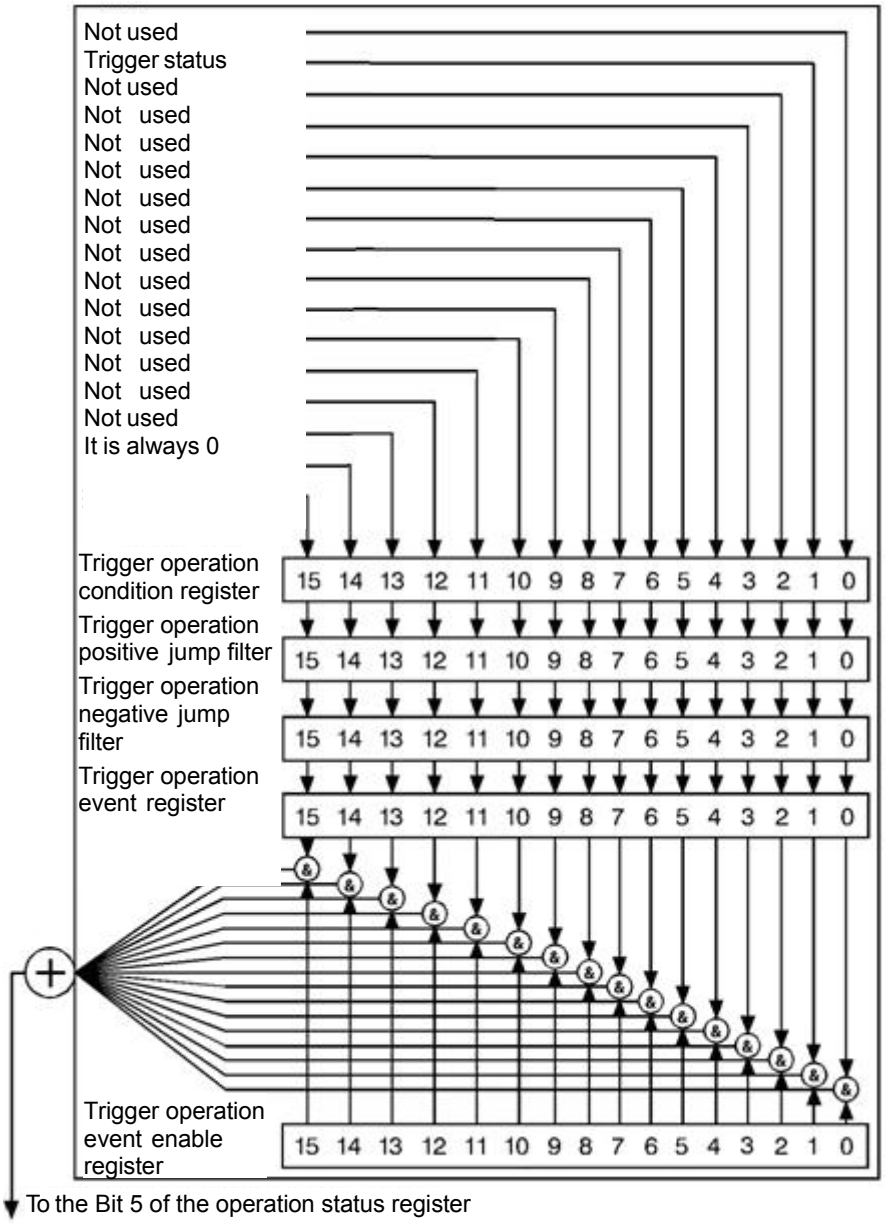


Fig.3.6 Trigger Operation Status Register

3.3.7.9 Summary of upper limit detection operation status register (STATus:OPERation:ULFail[:SUMMary])

Table 3.17 Summary of Upper Limit Detection Operation Status Register

Bit	Value	Definition
0-2	-	Not used
3	8	Upper limit detection status
4	16	Reserved
5	32	Reserved
6	64	Reserved
7-14	-	Not used
15	-	It is always 0

Bit 3 represents the failure detection status of the upper limit, and 1 means that it is out of the upper limit.

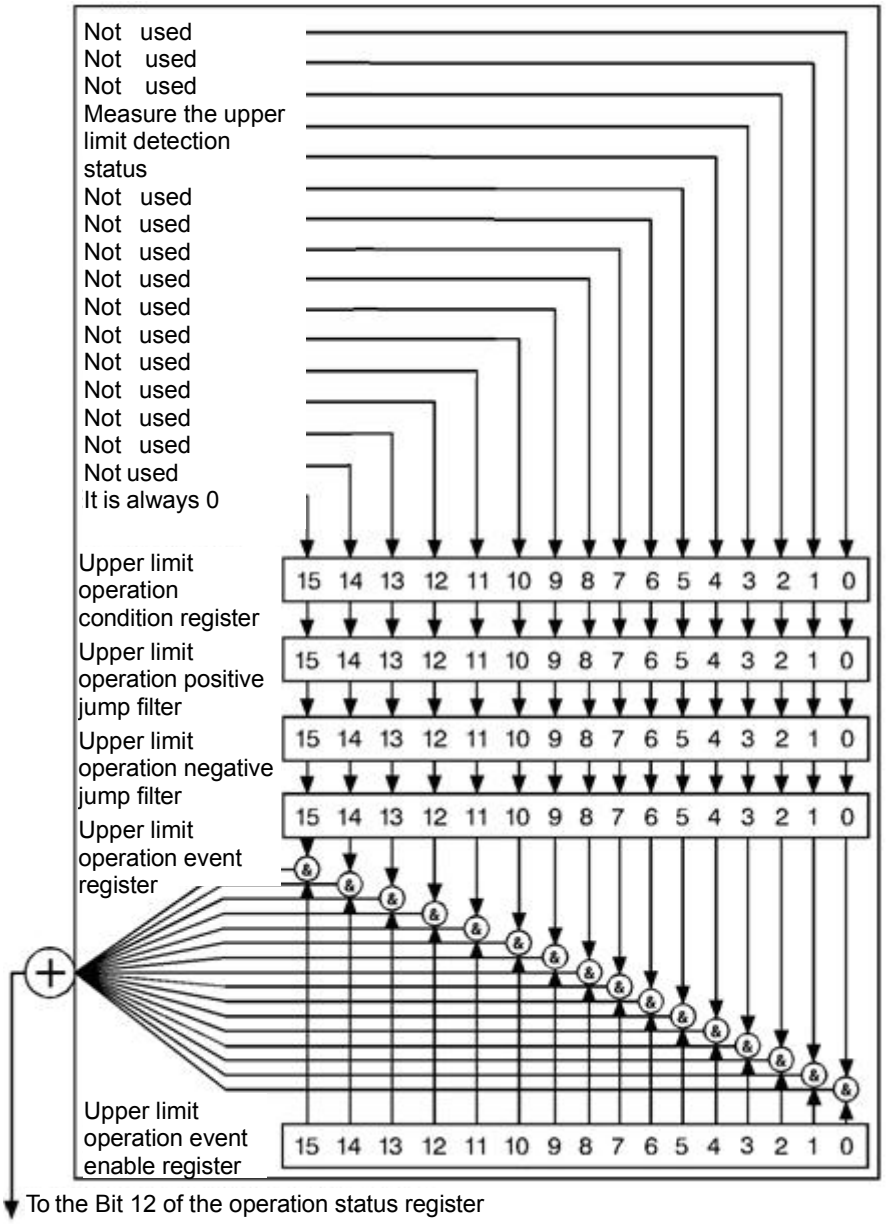


Fig.3.7 Upper Limit Detection Operation Status Register

3.3.7.10 Summary of questionable status register (STATus:OPERation:ULFail[:SUMMary])

Table 3.18 Summary of Questionable Status Register

Bit	Value	Definition
0-2	-	Not used
3	8	Power summary
4-7	-	Not used
8	256	Calibration summary
9	512	Power-on self-test
10-14	-	Not used
15	-	It is always 0

It includes 3 groups of operation registers:

- STATus:QUEStionable
- STATus:QUEStionable:Power[:SUMMary]
- STATus:QUEStionable:CALibration[:SUMMary]

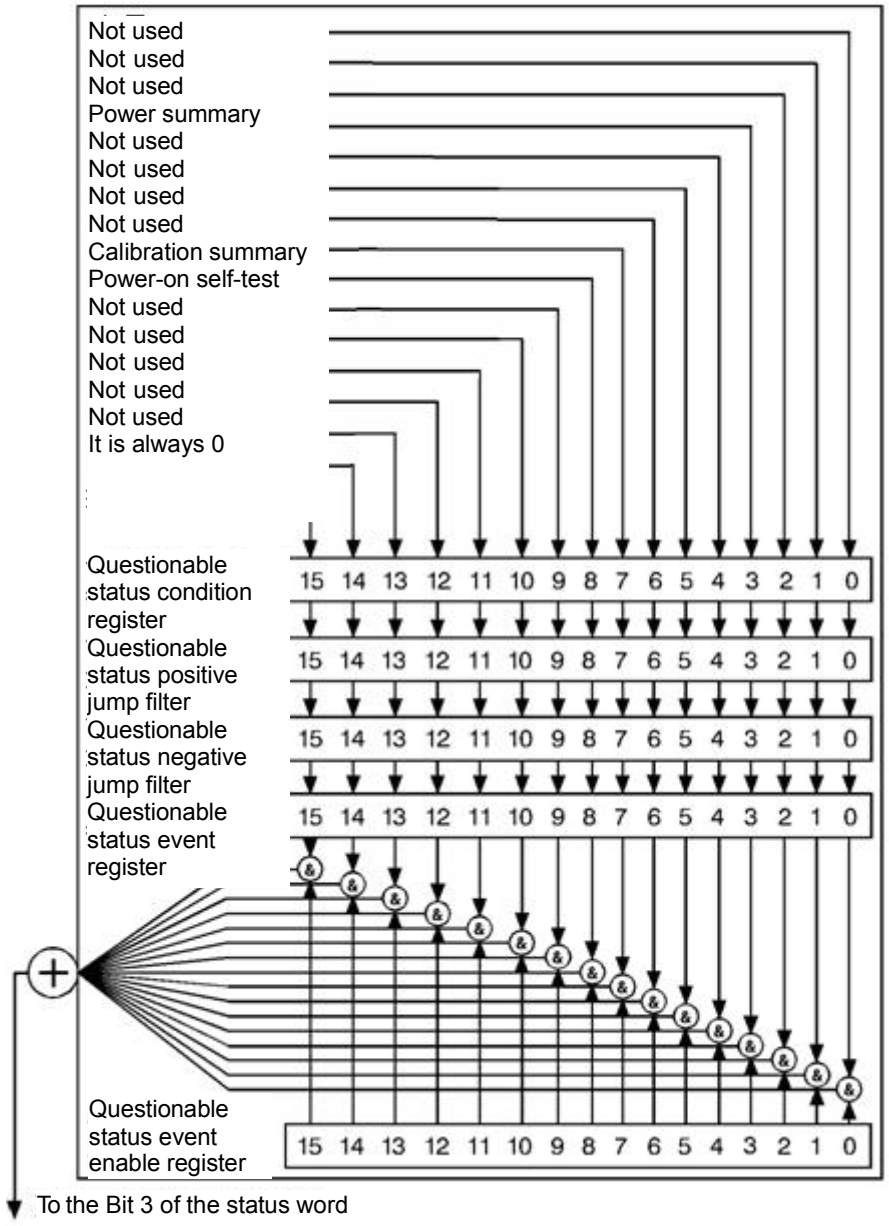


Fig.3.8 Questionable Status Register

3.3.7.11 Summary of calibration questionable status register (STATus:QUEStionable:CALibration[:SUMMary])

Table 3.19 Summary of Calibration Questionable Status Register

Bit	Value	Definition
0	1	Not used
1	2	Zero and calibration error
2	4	Reserved
3-14	-	Not used
15	-	It is always 0

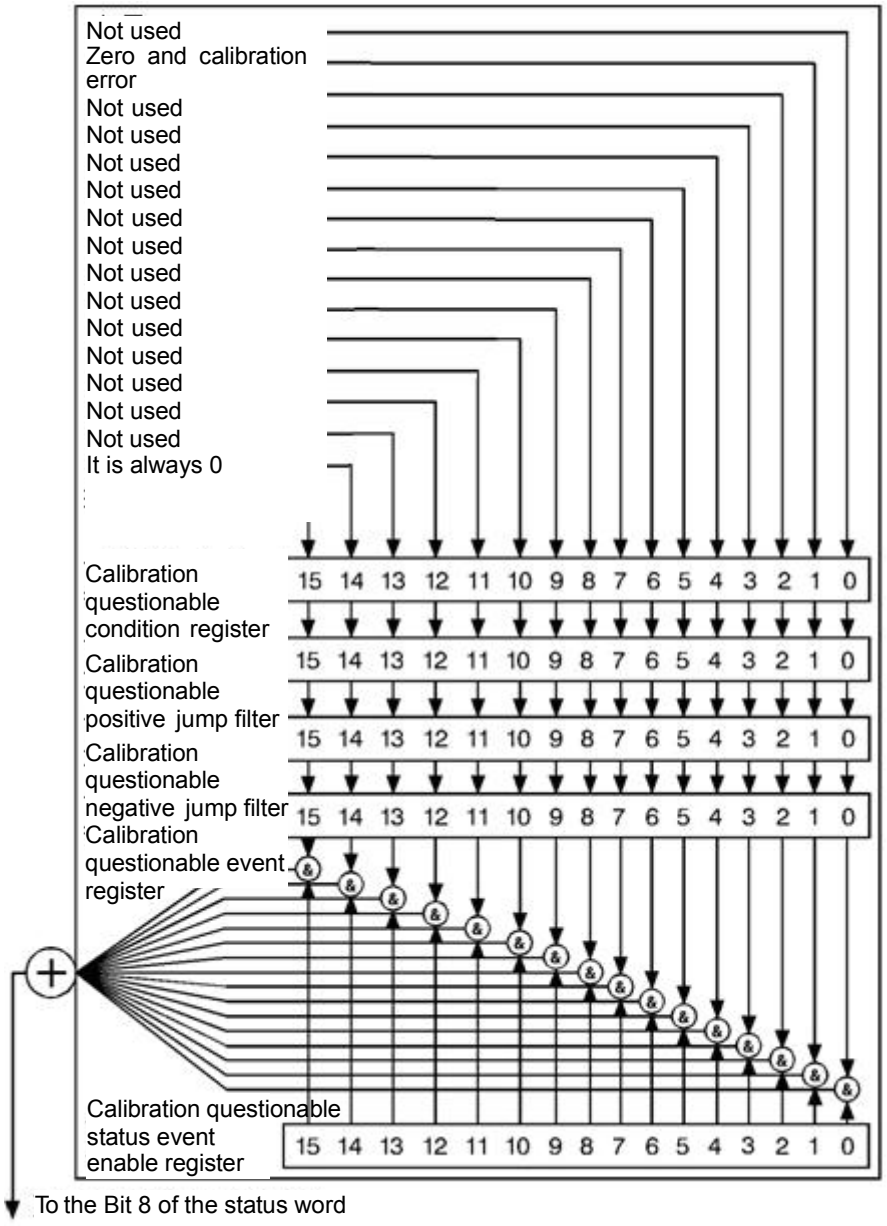


Fig.3.9 Calibration Questionable Status Register

3.3.7.12 Summary of power questionable status register (STATus:QUEStionable:POWer[:SUMMary])

Table 3.20 Summary of Power Questionable Status Register

Bit	Value	Definition
0	1	Not used
1	2	Input overload
2	4	Reserved
3	8	It shall be zeroed
4	16	Reserved
5	32	The data is invalid, or the log error occurs
6	64	Reserved
7	128	Reserved

8	256	Reserved
9-14	-	Not used
15	-	It is always 0

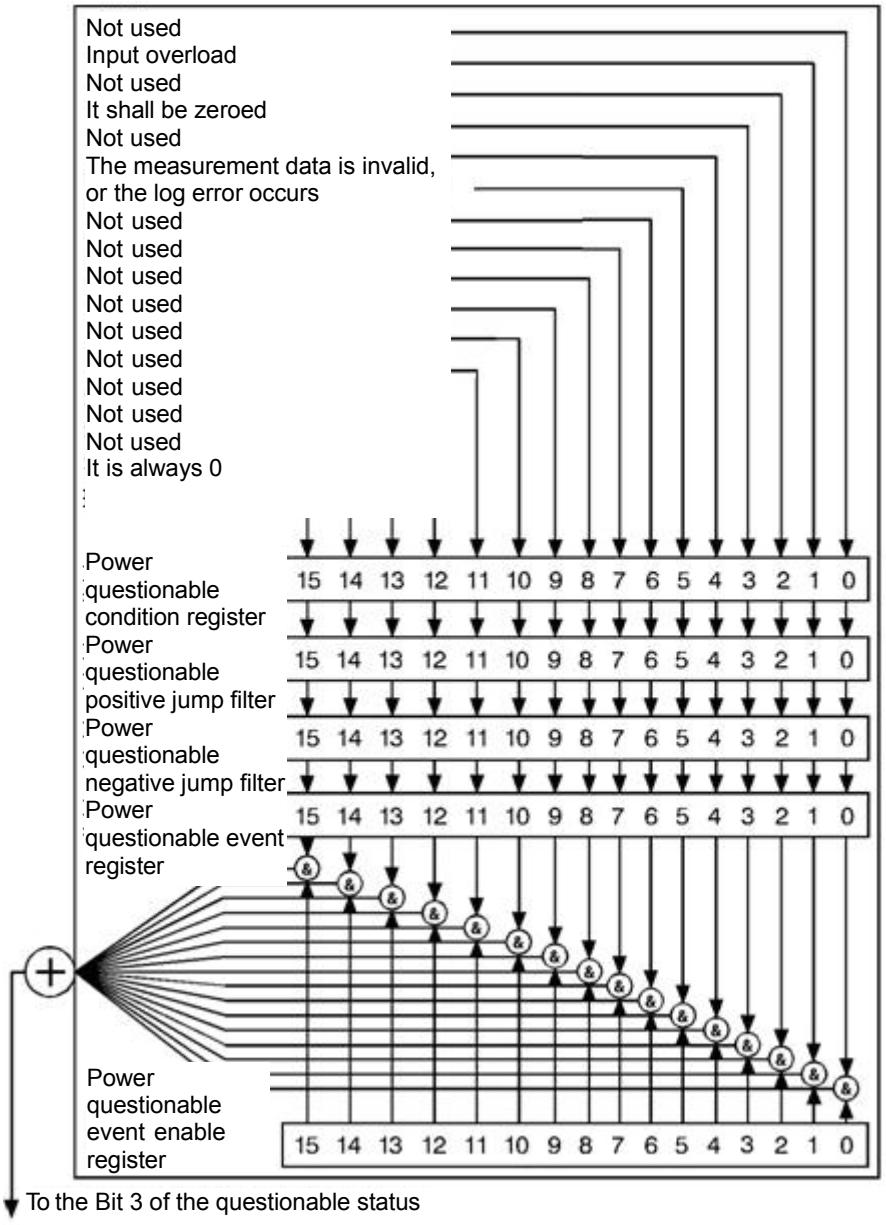


Fig.3.10 Power Questionable Status Register

:STATus:DEVice:CONDition?

Function: Query the value in the equipment status condition register.

For the returned value, if the Bit 1 is non-zero, it means that the sensor is detected; if the Bit 3 is non-zero, it means the sensor has error.

Query: :STATus:DEVice:CONDition?

Setting: Not supported

Example: STAT:DEV:COND?

STATus:DEVice:ENABle

Function: Query or set the equipment status event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit1 of the upper level status word. 1 represents enable.

Query: :STATus:DEVice:ENABle?

Setting: :STATus:DEVice:ENABle <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example:

STAT:DEV:ENAB?	Query the equipment status event enable register
STAT:DEV:ENAB 6	Enable the Bit 1 and Bit 2, namely, allow reporting the sensor detection event to the status word.
STAT:DEV:ENAB #B0110	The meaning is the same as above, and binary number is used.
STAT:DEV:ENAB #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:DEVice[:EVENt]?

Function: Query the equipment event register. The instrument will automatically reset the register after the query.

Query: :STATus:DEVice[:EVENt]?

Setting: Not supported

Example: STAT:DEV?

:STATus:DEVice:NTRansition

Function: Query or set the equipment negative jump filter.

Query: :STATus:DEVice:NTRansition?

Setting: :STATus:DEVice:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example:

STAT:DEV:NTR?	
STAT:DEV:NTR #H06	Report the status of the Bit 1 and Bit 2 of the negative jump filter to the event register.
STAT:DEV:NTR #B0110	The meaning is the same as above, and binary number is used.
STAT:DEV:NTR 6	The meaning is the same as above, and decimal number is used.

:STATus:DEVice:PTRansition

Function: Query or set the equipment positive jump filter.

Query: :STATus:DEVice:PTRansition?

Setting: :STATus:DEVice:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example:

STAT:DEV:PTR?	
STAT:DEV:PTR #H06	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:DEV:PTR #B0110	The meaning is the same as above, and binary number is used.

STAT:DEV:PTR 6 The meaning is the same as above, and decimal number is used.

:STATus:OPERation:CALibrating[:SUMMary]:CONDition?

Function: Query the value in the calibration status condition register.
For the returned value, if the Bit 1 is non-zero, it means that zero or calibration is being carried out. If 2 is returned, it means that it is being zeroed and calibrated.

Query: :STATus:OPERation:CALibrating[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:OPER:CAL:COND?

:STATus:OPERation:CALibrating[:SUMMary]:ENABle

Function: Query or set the calibration operation event enable register. Carry out bitwise-operation.
0 means that it is forbidden to report the status event to the Bit0 of the operation status.
1 represents enable.

Query: :STATus:OPERation:CALibrating[:SUMMary]:ENABle?

Setting: :STATus:OPERation:CALibrating[:SUMMary]:ENABle <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:CAL:ENAB?

STAT:OPER:CAL:ENAB 6	Enable the Bit 1 and Bit 2, namely, allow reporting the calibration operation event to the operation status.
STAT:OPER:CAL:ENAB #B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:CAL:ENAB #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:CALibrating[:SUMMary][:EVENt]?

Function: Query the calibration operation event register. The instrument will automatically reset the register after the query.

Query: :STATus:OPERation:CALibrating[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:OPER:CAL?

:STATus:OPERation:CALibrating[:SUMMary]:NTRansition

Function: Query or set the calibration operation negative jump filter.

Query: :STATus:OPERation:CALibrating[:SUMMary]:NTRansition?

Setting: :STATus:OPERation:CALibrating[:SUMMary]:NTRansition <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:CAL:NTR?

STAT:OPER:CAL:NTR? 6	Report the status of the Bit 1 and Bit 2 of the negative jump filter to the event register.

STAT:OPER:CAL:NTR?	#B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:CAL:NTR?	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:CALibrating[:SUMMary]:PTRansition

Function: Query or set the calibration operation positive jump filter.

Query: :STATus:OPERation:CALibrating[:SUMMary]:PTRansition?

Setting: :STATus:OPERation:CALibrating[:SUMMary]:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:CAL:PTR?

STAT:OPER:CAL:PTR?	6	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:OPER:CAL:PTR?	#B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:CAL:PTR?	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:CONDition?

Function: Query the value in the operation status condition register.

For the returned value, if the Bit 0 is non-zero, it means that the calibration event is detected; if the Bit 5 is non-zero, it means that the waiting trigger event is detected; if the Bit 10 is non-zero, it means that sensor connection or removal event is detected; if the Bit 11 is non-zero, it means that the lower limit detection event is detected; if the Bit 12 is non-zero, it means that the upper limit detection event is detected.

For example, if the calibration event is detected through this group of registers, the corresponding bit of the calibration enable register (STAT:OPER:CAL:ENAB) shall be set to non-zero. The others are similar.

Query: :STATus:OPERation:CONDition?

Setting: Not supported

Example: STAT:OPER:COND?

:STATus:OPERation:ENABle

Function: Query or set the operation status event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit7 of the status word. 1 represents enable.

Query: :STATus:OPERation:ENABle?

Setting: :STATus:OPERation:ENABle <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:CAL:ENAB?

STAT:OPER:CAL:ENAB?	1	Enable the Bit 1, namely, allow reporting the operation event to the status word.
STAT:OPER:CAL:ENAB?	#B0001	The meaning is the same as above, and binary number is used.

	STAT:OPER:CAL:ENAB? #H01	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:[:EVENt]?

Function: Query the operation status event register. The instrument will automatically reset the register after the query.

Query: :STATus:OPERation[:EVENt]?

Setting: Not supported

Example: STAT:OPER?

:STATus:OPERation:NTRansition

Function: Query or set the operation status negative jump filter.

Query: :STATus:OPERation:NTRansition?

Setting: :STATus:OPERation:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:NTR?

STAT:OPER:NTR 1	Report the status of the Bit 0 of the negative jump filter to the event register.
STAT:OPER:NTR #B0001	The meaning is the same as above, and binary number is used.
STAT:OPER:NTR #H01	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:PTRansition

Function: Query or set the operation status positive jump filter.

Query: :STATus:OPERation:PTRansition?

Setting: :STATus:OPERation:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:PTR?

STAT:OPER:PTR 6	Report the status of the Bit 0 of the positive jump filter to the event register.
STAT:OPER:PTR #B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:PTR #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:LLFail[:SUMMary]:CONDition?

Function: Query the value in the lower limit detection operation status condition register.

For the returned value, if the Bit 3 is non-zero, it represents the lower limit detection failure status.

Query: :STATus:OPERation:LLFail[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:OPER:LLF:COND?

:STATus:OPERation:LLFail[:SUMMary]:ENABle

Function: Query or set the lower limit detection operation event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit11 of the operation status. 1 represents enable.

Query: :STATus:OPERation:LLFail[:SUMMary]:ENABle?

Setting: :STATus:OPERation:LLFail[:SUMMary]:ENABle <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:LLF:ENAB?

STAT:OPER:LLF:ENAB	8	Enable the Bit 3, namely, allow reporting the lower detection operation event to the operation status.
STAT:OPER:LLF:ENAB	#B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:LLF:ENAB	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:LLFail[:SUMMary][:EVENt]?

Function: Query the lower limit detection operation event register. The instrument will automatically reset the register after the query.

Query: :STATus:OPERation:LLFail[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:OPER:LLF?

:STATus:OPERation:LLFail[:SUMMary]:NTRansition

Function: Query or set the lower detection operation negative jump filter.

Query: :STATus:OPERation:LLFail[:SUMMary]:NTRansition?

Setting: :STATus:OPERation:LLFail[:SUMMary]:NTRansition <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:LLF:NTR?

STAT:OPER:LLF:NTR	8	Report the status of the Bit 3 of the negative jump filter to the event register.
STAT:OPER:LLF:NTR	#B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:LLF:NTR	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:LLFail[:SUMMary]:PTRansition

Function: Query or set the lower detection operation positive jump filter.

Query: :STATus:OPERation:LLFail[:SUMMary]:PTRansition?

Setting: :STATus:OPERation:LLFail[:SUMMary]:PTRansition <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:LLF:PTR?

STAT:OPER:LLF:PTR 8	Report the status of the Bit 3 of the positive jump filter to the event register.
STAT:OPER:LLF:PTR #B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:LLF:PTR #H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:SENSe[:SUMMary]:CONDition?

Function: Query the value in the sense operation status condition register.
For the returned value, if the Bit 1 is non-zero, it represents the reading sensor EEPROM status.

Query: :STATus:OPERation:SENSe[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:OPER:SENS:COND?

:STATus:OPERation:SENSe[:SUMMary]:ENABle

Function: Query or set the sense operation event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the sense operation status event to the Bit10 of the operation status. 1 represents enable.

Query: :STATus:OPERation:SENSe[:SUMMary]:ENABle?

Setting: :STATus:OPERation:SENSe[:SUMMary]:ENABle <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:SENS:ENAB?

STAT:OPER:SENS:ENAB 6	Enable the Bit 1 and Bit 2, namely, allow reporting the sense operation event to the operation status.
STAT:OPER:SENS:ENAB #B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:SENS:ENAB #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:SENSe[:SUMMary][:EVENt]?

Function: Query the sense operation event register. The instrument will automatically reset the register after the query.

Query: :STATus:OPERation:SENSe[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:OPER:SENS?

:STATus:OPERation:SENSe[:SUMMary]:NTRansition

Function: Query or set the sense operation negative jump filter.

Query: :STATus:OPERation:SENSe[:SUMMary]:NTRansition?

Setting: :STATus:OPERation:SENSe[:SUMMary]:NTRansition <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT::OPER:SENS:NTR?

STAT:OPER:SENS:NTR 6	Report the status of the Bit 1 and Bit 2 of the negative jump filter to the event register.
STAT:OPER:SENS:NTR #B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:SENS:NTR #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:SENSe[:SUMMary]:PTRansition

Function: Query or set the sense operation positive jump filter.

Query: :STATus:OPERation:SENSe[:SUMMary]:PTRansition?

Setting: :STATus:OPERation:SENSe[:SUMMary]:PTRansition <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:SENS:NTR?

STAT:OPER:SENS:PTR 6	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:OPER:SENS:PTR #B0110	The meaning is the same as above, and binary number is used.
STAT:OPER:SENS:PTR #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:ULFail[:SUMMary]:CONDition?

Function: Query the value in the upper limit detection operation status condition register.
For the returned value, if the Bit 3 is non-zero, it represents the upper limit detection failure status.

Query: :STATus:OPERation:ULFail[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:OPER:ULF:COND?

:STATus:OPERation:ULFail[:SUMMary]:ENABle

Function: Query or set the upper limit detection operation event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit12 of the operation status. 1 represents enable.

Query: :STATus:OPERation:ULFail[:SUMMary]:ENABle?

Setting: :STATus:OPERation:ULFail[:SUMMary]:ENABle <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:OPER:ULF:ENAB?

STAT:OPER:ULF:ENAB 8	Enable the Bit 3, and allow reporting the upper detection operation event to the operation status.

STAT:OPER:ULF:ENAB	#B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:ULF:ENAB	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:ULFail[:SUMMary][:EVENt]?

Function: Query the upper limit detection operation event register. The instrument will automatically reset the register after the query.

Query: :STATus:OPERation:ULFail[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:OPER:ULF?

:STATus:OPERation:ULFail[:SUMMary]:NTRansition

Function: Query or set the upper detection operation negative jump filter.

Query: :STATus:OPERation:ULFail[:SUMMary]:NTRansition?

Setting: :STATus:OPERation:ULFail[:SUMMary]:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:ULF:NTR?

STAT:OPER:ULF:NTR	8	Report the status of the Bit 3 of the negative jump filter to the event register.
STAT:OPER:ULF:NTR	#B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:ULF:NTR	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:OPERation:ULFail[:SUMMary]:PTRansition

Function: Query or set the upper detection operation positive jump filter.

Query: :STATus:OPERation:ULFail[:SUMMary]:PTRansition?

Setting: :STATus:OPERation:ULFail[:SUMMary]:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:OPER:ULF:NTR?

STAT:OPER:ULF:PTR	8	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:OPER:ULF:PTR	#B1000	The meaning is the same as above, and binary number is used.
STAT:OPER:ULF:PTR	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:PRESet

Function: Preset some status registers as follows, and other registers will be kept unchanged.

Query: Not supported

Setting: :STATus:PRESet

Example: STAT:PRES

Table 3.21 Summary of Reset Status of Status Register

Register	Sub-register	Preset Status
OPERation	ENABle	All is set to 0
	PTR	All is set to 1
	NTR	All is set to 0
QUEStionable	ENABle	All is set to 0
	PTR	All is set to 1
	NTR	All is set to 0
Others	ENABle	All is set to 0
	PTR	All is set to 1
	NTR	All is set to 0

:STATus:QUEStionable:CALibration[:SUMMary]:CONDition?

Function: Query the value in the calibration questionable status condition register.
For the returned value, if the Bit 1 is non-zero, it represents zero and calibration error. For example, if 2 is returned, it represents zero and calibration error.

Query: :STATus:OPERation:CALibrating[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:OPER:CAL:COND?

:STATus:QUEStionable:CALibration[:SUMMary]:ENABle

Function: Query or set the calibration questionable event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit8 of the questionable status. 1 represents enable.

Query: :STATus:QUEStionable:CALibration[:SUMMary]:ENABle?

Setting: :STATus:QUEStionable:CALibration[:SUMMary]:ENABle <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example: STAT:QUES:CAL:ENAB?

STAT:QUES:CAL:ENAB	6	Enable the Bit 1 and Bit 2, and allow reporting the calibration questionable event to the questionable status.
STAT:QUES:CAL:ENAB	#B0110	The meaning is the same as above, and binary number is used.
STAT:QUES:CAL:ENAB	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:CALibration[:SUMMary][:EVENt]?

Function: Query the calibration questionable event register. The instrument will automatically reset the register after the query.

Query: :STATus:QUEStionable:CALibration[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:QUES:CAL?

:STATus:QUEStionable:CALibration[:SUMMary]:NTRansition

Function: Query or set the calibration questionable negative jump filter.

Query: :STATus:QUEStionable:CALibration[:SUMMary]:NTRansition?

Setting: :STATus:QUEStionable:CALibration[:SUMMary]:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:CAL:NTR?

STAT:QUES:CAL:NTR	6	Report the status of the Bit 1 and Bit 2 of the negative jump filter to the event register.
STAT:QUES:CAL:NTR	#B0110	The meaning is the same as above, and binary number is used.
STAT:QUES:CAL:NTR	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:CALibration[:SUMMary]:PTRansition

Function: Query or set the calibration questionable positive jump filter.

Query: :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition?

Setting: :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:CAL:PTR?

STAT:QUES:CAL:PTR	6	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:QUES:CAL:PTR	#B0110	The meaning is the same as above, and binary number is used.
STAT:QUES:CAL:PTR	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:CONDition?

Function: Query the value in the questionable status condition register.

For the returned value, if the Bit 3 is non-zero, it means that the power questionable event is detected; if the Bit 8 is non-zero, it means that the calibration questionable event is detected; if the Bit 9 is non-zero, it means that the power-on self-test fails.

For example, if the calibration questionable event is detected through this group of registers, the corresponding bit of the calibration questionable enable register (STAT:QUES:CAL:ENAB) shall be set to non-zero. The others are similar.

Query: :STATus:QUEStionable:CONDition?

Setting: Not supported

Example: STAT:QUES:COND?

:STATus:QUEStionable:ENABle

Function: Query or set the questionable status event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit3 of the questionable status. 1 represents enable.

Query: :STATus:QUEStionable:ENABle?

Setting: :STATus:QUEStionable:ENABle <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:ENAB?

STAT:QUES:ENAB	8	Enable the Bit 1, namely, allow reporting the calibration questionable event to the status word.
STAT:QUES:ENAB	#B1000	The meaning is the same as above, and binary number is used.
STAT:QUES:ENAB	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable[:EVENt]?

Function: Query the questionable status event register. The instrument will automatically reset the register after the query.

Query: :STATus:QUEStionable[:EVENt]?

Setting: Not supported

Example: STAT:QUES?

:STATus:QUEStionable:NTRansition

Function: Query or set the questionable status negative jump filter.

Query: :STATus:QUEStionable:NTRansition?

Setting: :STATus:QUEStionable:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:NTR?

STAT:QUES:NTR	8	Report the status of the Bit 3 of the negative jump filter to the event register.
STAT:QUES:NTR	#B1000	The meaning is the same as above, and binary number is used.
STAT:QUES:NTR	#H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:PTRansition

Function: Query or set the questionable status positive jump filter.

Query: :STATus:QUEStionable:PTRansition?

Setting: :STATus:QUEStionable:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example:	STAT:QUES:PTR?	
	STAT:QUES:PTR 8	Report the status of the Bit 3 of the positive jump filter to the event register.
	STAT:QUES:PTR #B0110	The meaning is the same as above, and binary number is used.
	STAT:QUES:PTR #H08	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:POWer[:SUMMary]:CONDition?

Function: Query the value in the power questionable status condition register. For the returned value,
if the Bit 1 is non-zero, it represents input overload;
if the Bit 3 is non-zero, it means that it shall be zeroed;
if the Bit 5 is non-zero, it means that the data is invalid, or the log error occurs;
if 8 is returned, it means that it shall be zeroed.

Query: :STATus:QUEStionable:POWer[:SUMMary]:CONDition?

Setting: Not supported

Example: STAT:QUES:POW:COND?

:STATus:QUEStionable:POWer[:SUMMary]:ENABle

Function: Query or set the power questionable event enable register. Carry out bitwise-operation. 0 means that it is forbidden to report the status event to the Bit3 of the questionable status. 1 represents enable.

Query: :STATus:QUEStionable:POWer[:SUMMary]:ENABle?

Setting: :STATus:QUEStionable:POWer[:SUMMary]:ENABle? <NRf>|<Non-decimal>
The range of the parameter is 0 - 32767.

Example:	STAT:QUES:POW:ENAB?	
	STAT:QUES:POW:ENAB 6	Enable the Bit 1 and Bit 2, namely, allow reporting the power questionable event to the questionable status.
	STAT:QUES:POW:ENAB #B0110	The meaning is the same as above, and binary number is used.
	STAT:QUES:POW:ENAB #H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:POWer[:SUMMary][:EVENt]?

Function: Query the power questionable event register. The instrument will automatically reset the register after the query.

Query: :STATus:QUEStionable:POWer[:SUMMary][:EVENt]?

Setting: Not supported

Example: STAT:QUES:POW?

:STATus:QUEStionable:POWer[:SUMMary]:NTRansition

Function: Query or set the power questionable negative jump filter.

Query: :STATus:QUEStionable:POWer[:SUMMary]:NTRansition?

Setting: :STATus:QUEStionable:POWer[:SUMMary]:NTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:POW:NTR?

STAT:QUES:POW:NTR	6	Report the status of the Bit 1 and Bit 2 of the negative jump filter to the event register.
STAT:QUES:POW:NTR	#B0110	The meaning is the same as above, and binary number is used.
STAT:QUES:POW:NTR	#H06	The meaning is the same as above, and hexadecimal number is used.

:STATus:QUEStionable:POWer[:SUMMary]:PTRansition

Function: Query or set the power questionable positive jump filter.

Query: :STATus:QUEStionable:POWer[:SUMMary]:PTRansition?

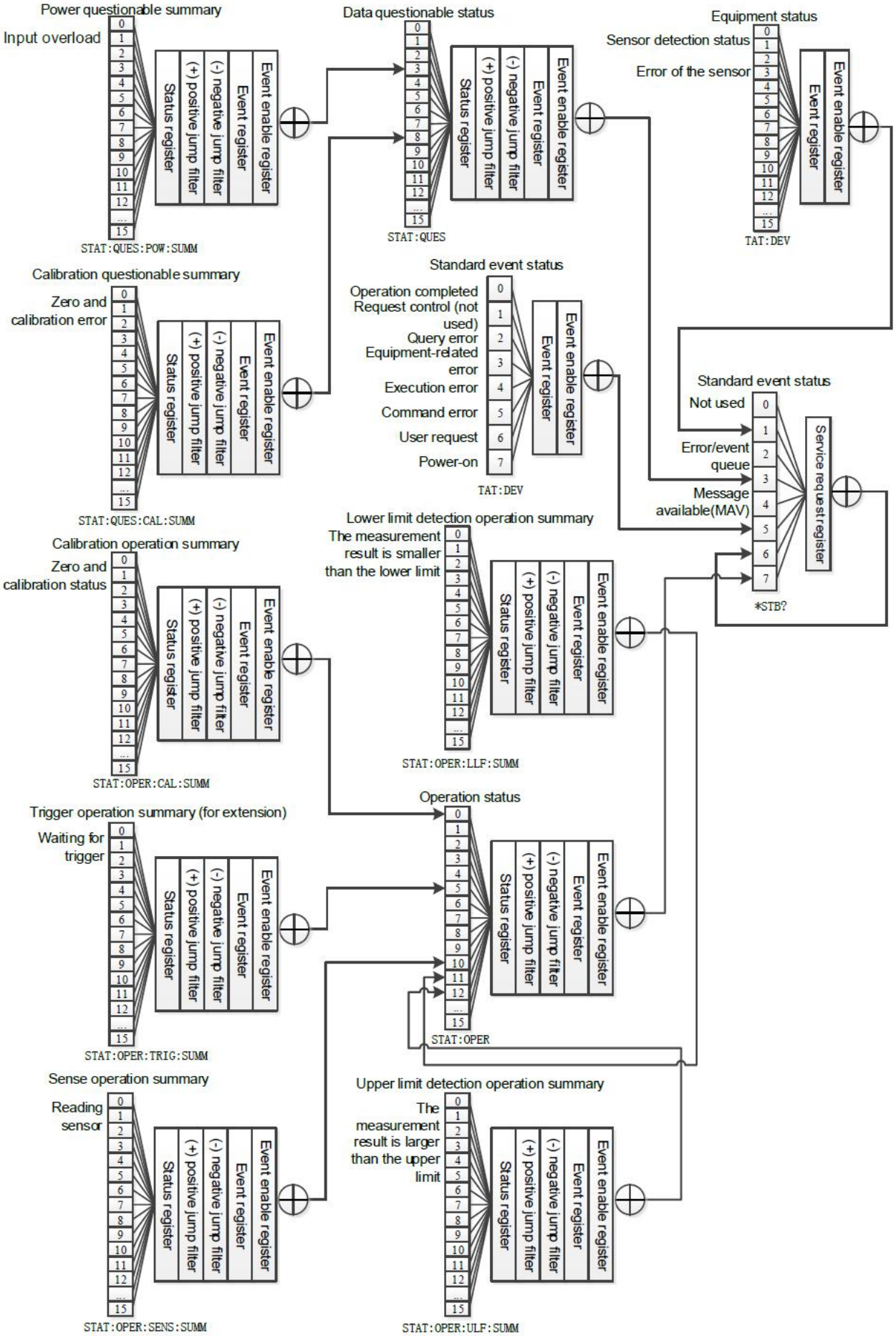
Setting: :STATus:QUEStionable:POWer[:SUMMary]:PTRansition <NRf>|<Non-decimal>

The range of the parameter is 0 - 32767.

Example: STAT:QUES:POW:PTR?

STAT:QUES:POW:PTR	6	Report the status of the Bit 1 and Bit 2 of the positive jump filter to the event register.
STAT:QUES:POW:PTR	#B0110	The meaning is the same as above, and binary number is used.
STAT:QUES:POW:PTR	#H06	The meaning is the same as above, and hexadecimal number is used.

Attached: Status Block Diagram


Power questionable summary
Input overload
Status register
(+) positive jump filter
(-) negative jump filter
Event register
Event enable register
STAT:QUES:POW:SUMM
Data questionable status
STAT:QUES
Equipment status
Sensor detection status
Error of the sensor
TAT:DEV
Calibration questionable summary
Zero and calibration error
STAT:QUES:CAL:SUMM
Standard event status
Operation completed
Request control (not used)
Query error
Equipment-related error
Execution error
Command error
User request
Power-on
TAT:DEV
Standard event status
Not used
Error/event queue
Message available(MAV)
Service request register
*STB?
Lower limit detection operation summary
The measurement result is smaller than the lower limit
STAT:OPER:LLF:SUMM
Calibration operation summary
Zero and calibration status
STAT:OPER:CAL:SUMM
Operation status
STAT:OPER
Trigger operation summary (for extension)
Waiting for trigger
STAT:OPER:TRIG:SUMM
Sense operation summary
Reading sensor
STAT:OPER:SENS:SUMM
Upper limit detection operation summary
The measurement result is larger than the upper limit
STAT:OPER:ULF:SUMM

3.3.8 System (SYSTem)

- :SYSTem:ERRor[:NEXT]?
- :SYSTem:HELP:HEADers?
- :SYSTem:PRESet
- :SYSTem:VERSion?

:SYSTem:ERRor[:NEXT]?

Function: Return the error code and error information of the instrument from its error queue. When an error occurs, the error code and error information can be saved in the error queue. After the command is executed once, the information in the error queue will be removed. The order removing the error information from the queue is FIFO, namely, the oldest information will be removed from the queue first. The *CLS command can be used to clear the error queue. When the error queue is null and the command is executed, “0, "No Error"” will be returned. The error queue can contain up to 1023 pieces of error information.

Query: :SYSTem:ERRor[:NEXT]?

Setting: Not supported

Example: SYST:ERR?

Reset status: No impact.

:SYSTem:HELP:HEADers?

Function: Query the command list supported by the instrument. The format of the data is <Any Data Block> See section 7.7.6 of the IEEE 488.2. #nNNN...Nddd……ddd<LF>

Query: :SYSTem:HELP:HEADers?

Setting: Not supported

Example: SYST:HELP:HEAD?

:SYSTem:PRESet

Function: Reset the instrument

Query: Not supported

Setting: :SYSTem:PRESet [Character Data]

The form of [Character Data] is as follows (DEFault is a default parameter, namely the parameter without parameters.)

Example: SYST:PRES DEF Reset

Reset status:

:SYSTem:VERSion?

Function: Query SCPI version number of the instrument. The returned form is YYYY.X, YYYY represents year and X represents version number. 1999.0 will be returned in this case.

Query: :SYSTem:VERSion?

Settings: Not supported

Example: SYST:VERS?

3.3.9 Trigger (INITiate/TRIGger)

- ABORt[1]
- :INITiate[1]:CONTinuous
- :INITiate:CONTinuous:ALL
- :INITiate:CONTinuous:SEQuence[1]
- :INITiate[1][:IMMediate]
- :INITiate[:IMMediate]:ALL
- :INITiate[:IMMediate]:SEQuence[1]
- :TRIGger[1][:IMMediate

:ABORt[1]

Function: Disable measurement. Restart measurement after sending INITiate:CONTinuous ON.

Query: Not supported

Setting: ABORt[1]

Example: ABOR Disable measurement.

Description: The instrument doesn't have the trigger idle status, namely, the trigger is always in the waiting status.

:INITiate[1]:CONTinuous

Function: Query or set the trigger status of the instrument: Single step and continuous
When it is set to single step (OFF), wait for trigger until it is set to continuous (ON) or INITiate:IMMediate is received.
The command is equivalent to INITiate:CONTinuous:SEQuence[1].

Query: :INITiate[1]:CONTinuous?

Setting: :INITiate[1]:CONTinuous <Boolean Data>
OFF|0 Single step
ON|1 Continuous

Example: INIT:CONT? Query the trigger status, 0 single step; 1 continuous.
INIT:CONT ON Set the continuous trigger status.

Description: For the CW sensor, it can't be set to single step trigger and continuous trigger; for the parameter OFF, it is equivalent to stopping the measurement, and for the parameter ON, it is equivalent to starting the measurement.

Reset status: Set it to ON.

:INITiate:CONTinuous:ALL

Function: Query or set the trigger status of all channels of the instrument: Single step and continuous
When it is set to single step (OFF), wait for trigger until it is set to continuous (ON) or INITiate:IMMediate is received.

Query: :INITiate:CONTinuous:ALL?

Setting: :INITiate:CONTinuous:ALL <Boolean Data>
OFF|0 Single step
ON|1 Continuous

Example: INIT:CONT:ALL? Query the trigger status of all the channels. When all channels are continuously triggered, it will be 1, otherwise, it will be 0.

INIT:CONT:ALL ON Set all the channels to the continuous trigger status.

Reset status: Set it to ON

:INITiate:CONTinuous:SEQuence[1]

Function: Query or set the trigger status of the instrument: Single step and continuous
When it is set to single step (OFF), wait for trigger until it is set to continuous (ON) or INITiate:IMMediate is received.
The command is equivalent to INITiate[1]:CONTinuous

Query: :INITiate:CONTinuous:SEQuence

Setting: :INITiate:CONTinuous:SEQuence[1] <Boolean Data>
OFF|0 Single step
ON|1 Continuous

Example: INIT:CONT:SEQ? Query the trigger status, 0 single step; 1 continuous.

INIT:CONT:SEQ1 ON Set the continuous trigger status.

:INITiate[1][:IMMediate]

Function: Set the instrument to the waiting for trigger status. The measurement will start when trigger event is received.
The equivalent command: INITiate:[IMMediate]:SEQuence[1]

Query: Not supported

Setting: :INITiate[1][:IMMediate]

Example: INIT Set it to the waiting for trigger status.

:INITiate[:IMMediate]:ALL

Function: Set all channels of the instrument to the waiting for trigger status. The measurement will start when trigger event is received.

Query: Not supported

Setting: :INITiate[:IMMediate][:ALL]

Example: INIT:ALL Set all channels of the instrument to the waiting for trigger status.

:INITiate[:IMMediate]:SEQuence[1]

Function: Set the instrument to the waiting for trigger status. The measurement will start when trigger event is received.
The equivalent command :INITiate[1][:IMMediate]

Query: Not supported

Setting:	:INITiate[:IMMediate]:SEQuence[1]	
Example:	INIT:SEQ	Set it to the waiting for trigger status.

:TRIGger[1][:IMMediate]

Function:	The command sets the instrument to the trigger waiting status immediately. The equivalent command: INITiate[1]:[IMMediate] TRIGger[:SEQuence[1]][:IMMediate]	
Query:	Not supported	
Setting:	:TRIGger[1][:IMMediate]	
Example:	TRIG	It sets the instrument to the trigger waiting status immediately.

3.3.10 Unit (UNIT)

- :UNIT[1]:POWer
- :UNIT[1]:POWer:RATio

:UNIT[1]:POWer

Function:	Query or set the power unit. The linearity and logarithm are in the menu operation. The absolute power measurement unit is W and dBm, which correspond to linearity and logarithm respectively. The ratio measurement and relative power measurement unit are % and dB, which correspond to linearity and logarithm respectively.	
Query:	:UNIT[1]:POWer?	
Setting:	:UNIT[1]:POWer <Character Data> The valid character data includes: DBM or 0: Log display. W or 1: Linear display	
Example:	UNIT:POW?	Query the power unit.
	UNIT1:POW W	Set the power unit to W.

:UNIT[1]:POWer:RATio

Function:	Query or set the ratio measurement power unit. The linearity and logarithm are in the menu operation.	
Query:	:UNIT[1]:POWer:RATio?	
Setting:	:UNIT[1]:POWer:RATio<Character Data> The valid character data includes: DB or 0: Log display. PCT or 1: Linear display (PCT represents %)	
Example:	UNIT:POW:RAT?	Query the power unit.
	UNIT1:POW:RAT PCT	Set the power unit to PCT.

3.3.11 Service (SERVice)

- :SERVice:LANGuage
- :SERVice:SENSor[1]:CDATe?

- :SERVice:SENSor[1]:CPLace?
- :SERVice:SENSor[1]:FREQuency:MAXimum?
- :SERVice:SENSor[1]:FREQuency:MINimum?
- :SERVice:SENSor[1]:SNUMber?
- :SERVice:SENSor[1]:TYPE?
- :SERVice:SNUMber

:SERVice:LANGuage

Function: Query or set the language selection.

Query: :SERVice:LANGuage?

Setting: :SERVice:LANGuage <CHINese|ENGLish|0|1>

CHINese or 0: Select Chinese

ENGLish or 1: Select English

Example: SERV:LANG? Query the language selection.

SERV:LANG CHIN Set the language to Chinese

Reset status: No impact.

:SERVice:SENSor[1]:CDATe?

Function: Query the sensor calibration date. The calibration date is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:CDATe?

Setting: Not supported

Example: SERV:SENS:CDAT? Query the sensor calibration date.

:SERVice:SENSor[1]:CPLace?

Function: Query the sensor calibration address. The calibration address is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:CPLace?

Setting: Not supported

Example: SERV:SENS:CPL? Query the sensor calibration address.

:SERVice:SENSor[1]:FREQuency:MAXimum?

Function: Query the maximum frequency. The maximum frequency is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:FREQuency:MAXimum?

Setting: Not supported

Example: SERV:SENS:FREQ:MAX? Query the maximum frequency of the sensor.

:SERVice:SENSor[1]:FREQuency:MINimum?

Function: Query the minimum frequency. The minimum frequency is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:FREQuency:MINimum?

Setting: Not supported

Example: SERV:SENS:FREQ:MIN? Query the minimum frequency of the sensor.

:SERVice:SENSor[1]:SNUMber?

Function: Query the serial number. The serial number of the sensor is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:SNUMber?

Setting: Not supported

Example: SERV:SENS:SNUM? Query the serial number of the sensor.

:SERVice:SENSor[1]:TYPE?

Function: Query the type. The type of the sensor is saved in the sensor EEPROM.

Query: :SERVice:SENSor[1]:TYPE?

Setting: Not supported

Example: SERV:SENS:TYPE? Query the type of the sensor.

:SERVice:SNUMber

Function: Query or set the serial number of the instrument.

Query: :SERVice:SNUMber?

Setting: :SERVice:SNUMber <Character Data> It will be kept temporarily.

Example: SERV:SNUM?

Chapter 4 Programming Example

4.1 Basic Operation Example

The basic methods for programming of remote control of the instrument through the VISA library are illustrated hereinafter. Take the C++ language as an example.

- VISA library
- Example Running Environment
- Initialization and Default Status Setting
- Sending of Setting Command
- Reading of Instrument Status
- Synchronization of Command

4.1.1 VISA Library

The VISA is a general term of the standard I/O function library and its relevant specifications. The VISA library function is a set of functions that can be easily recalled. Its core function can control different types of devices without considering the interface types of devices and the operation methods of different sets of I/O interface software. These library functions are used to write the driver of the instrument as well as complete the command and data transmission between the computer and the instrument, so as to realize the remote control of the instrument. The instruments with remote interfaces (LAN, USB, GPIB and RS-232) can be connected through initializing the addressing string ("VISA Resource String").

At first, it is necessary to install the VISA library so as to achieve remote control. The VISA library packages the underlying transfer functions of underlying VXI, GPIB, LAN and USB interfaces so that the user can recall them directly. The programming interface supported by this instrument is: USB. These interfaces can be used together with the VISA library and programming language for remote control of the instrument. Currently, Keysight I/O Library is often used as the underlying I/O library.

Figure 4.1 shows the relationship between remote interface, VISA library, programming language and the power sensor taking USB interface as an example.

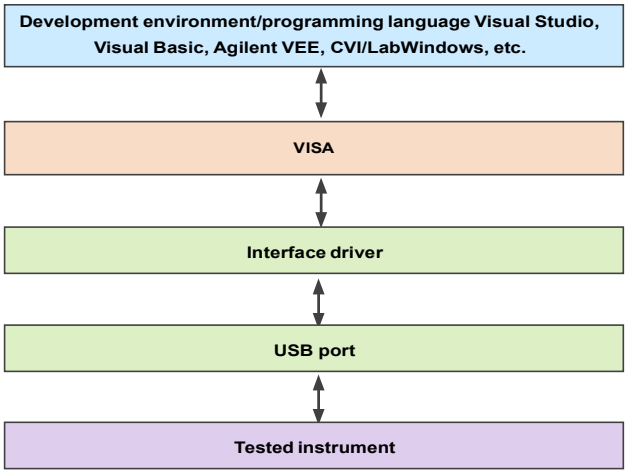


Fig.4.1 Remote control hardware and software layer

4.1.2 Example Running Environment

4.1.2.1 Configuration requirements

The programming examples described in this chapter have been successfully run on the computers with the following configuration.

- IBM compatible, Pentium PC and above;
- Windows 2000, Windows XP, Windows7 or Windows8 and Windows10 operating systems;
- Visual Studio 2010/2012 integrated development environment;
- NI VISA library or Keysight VISA library.

4.1.2.2 Included files

To run C/C++ program example, you shall include the necessary files in VC project If you use VISA library, you must perform the following steps:

- Add visa32.lib to the source file;
- Add visa.h to the header file.

For more information about VISA library, please browse in the websites of NI company and Keysight company respectively.

4.1.3 Initialization and Default Status Setting

When the program starts, firstly initialize VISA resource manager, and then enable and establish the communication connection between VISA library and the instrument. The specific steps are as follows:

4.1.3.1 Generation of global variable

Firstly, generate the global variable to be recalled by other program modules, such as the instrument handle variable. The following example program shall contain the following global variables:

```
ViSession iDevHandle;
ViSession iDefaultRM;
const char rgcDevResource[VI_FIND_BUFLEN] = "USB0::1204::4100::1803001::0::INSTR";
const analyzerTimeout = 5000;
```

The constant rgcDevResource represents the instrument descriptor.

4.1.3.2 Controller initialization

```
/*************************************************************************
```

The following example shows how to open and establish a communication connection between VISA library and the instrument (specified by instrument descriptor).

```
Controller initialization: Turn on the default resource manager and return the instrument handle iDevHandle.
*************************************************************************/
    void InitController()
    {
    ViStatus iStatus;
        iStatus = viOpenDefaultRM(&iDefaultRM);
        iStatus = viOpen(iDefaultRM, rgcDevResource, VI_NULL, VI_NULL, &iDevHandle);
```

```
}
```

4.1.3.3 Instrument initialization

/**/

The following example shows how to initialize the instrument default status and clear status registers.

/**/

```
void InitDevice()
{
ViStatus iStatus;
    ViUInt32 uiRetCnt;
    iStatus = viWrite(iDevHandle, "*CLS\n", strlen("*CLS\n"), &uiRetCnt); //reset status register
    iStatus = viWrite(iDevHandle, "*RST\n", strlen("*RST\n"), &uiRetCnt); //reset instrument
}
```

4.1.4 Sending of Setting Command

/**/

The following example describes how to set the frequency of S87230.

/**/

```
void SimpleSettings()
{
ViStatus iStatus;
    ViUInt32 uiRetCnt;
    //Set the frequency as 128 MHz
    iStatus = viWrite(iDevHandle, "FREQ 1.28e8\n", strlen("FREQ 1.28e8\n"), &uiRetCnt);
}
```

4.1.5 Reading of Instrument Status

/**/

The following example shows how to read the setting status of the instrument.

/**/

```
void ReadSettings()
{
ViStatus iStatus;
    ViUInt32 uiRetCnt;
    char rgcBuf[256];

    //Query the frequency
    iStatus = viWrite(iDevHandle, "FREQ?\n", strlen("FREQ?\n"), &uiRetCnt);
    Sleep(10);
    iStatus = viRead(iDevHandle, rgcBuf, sizeof(rgcBuf), &uiRetCnt);
    //Print debugging information
```

```
    printf("Frequency is %s", rgcBuf);
}
```

4.1.6 Synchronization of command

```
/*****************************************************************************/
```

The methods for command synchronization are illustrated hereinafter by taking sweep as an example.

```
/*****************************************************************************/
 void SweepSync()
 {
      ViStatus iStatus;
           ViUInt32
           uiRetCnt;
           ViEventType
           eType;
           ViEvent
           eEvent;
           int iStat;
           char rgcOpcOk[2];

           /*****************************************************************/
           /* Command INITiate[:IMMediate]    Start a single scanning (when continuous scanning is
           disabled INIT:CONT OFF)*/
           /* When a single scanning is completed, next command in the command buffer area can be
           executed.*/
           /*****************************************************************
           ******/ iStatus = viWrite(iDevHandle, "INIT:CONT OFF", 13, &uiRetCnt);
           //Method 1 of waiting for sweep completion: use *WAI
           iStatus = viWrite(iDevHandle, "ABOR;INIT:IMM;*WAI", 18, &uiRetCnt);
           //Method 2 of waiting for sweep completion: use *OPC?
           iStatus = viWrite(iDevHandle, "ABOR;INIT:IMM; *OPC?", 20, &uiRetCnt);
           iStatus = viRead(iDevHandle, rgcOpcOk, 2, &uiRetCnt); //wait for *OPC to return "1"
           //Method 3 of waiting for sweep completion: use *OPC
           //To use GPIB service request, set "Disable Auto Serial Poll" to "yes"
           iStatus = viWrite(iDevHandle, "*SRE 32", 7, &uiRetCnt);
           iStatus = viWrite(iDevHandle, "*ESE 1", 6, &uiRetCnt); //enable service request ESR
           //Set the event enable bit, the operation is completed
           iStatus = viEnableEvent(iDevHandle, VI_EVENT_SERVICE_REQ, VI_QUEUE, VI_NULL);
           //Enable SRQ event
           iStatus = viWrite(iDevHandle, "ABOR;INIT:IMM;*OPC", 18, &uiRetCnt);
           //Start sweep synchronously with OPC
           iStatus = viWaitOnEvent(iDevHandle, VI_EVENT_SERVICE_REQ, 10000, &eType, &eEvent)
           //Wait for service request
```

```
        iStatus = viReadSTB(iDevHandle, &iStat);
        iStatus = viClose(eEvent); //close the event handle
        //Disable the SRQ event
        iStatus = viDisableEvent(iDevHandle, VI_EVENT_SERVICE_REQ, VI_QUEUE);
        //The main program continues……
}
```

4.2 Advanced Operation Example

- USBTMC Remote Control Example

4.2.1 USBTMC Remote Control Example

4.2.1.1 Before using the example

Install the VISA library and the USBTMC equipment correctly. The process of recalling the VISA library is as follows:

1) Open the default resource manager with viOpenDefaultRM to obtain the resource management handle defaultRM.
2) Turn on the equipment, and there will be two conditions.

a. The resource address of the controlled instrument (equipment) is known.

Turn on the instrument in the specified resource address with viOpen to obtain the session handle vi of the instrument. The example is as shown in UsbTest1.

b. The resource address of the controlled instrument (equipment) is unknown.

Find the available equipment resource with viFindRsrc and viFindNext; turn on the instrument in the specified resource address with viOpen to obtain the session handle vi of the instrument. The example is as shown in UsbTest2.

3) Carry out write operation to the equipment with functions including viPrintf and viWrite, carry out read operation to the equipment with functions including viScanf and viRead, and carry out read-write operation to the equipment with functions including viQueryf.
4) Close the instrument handle and resource management handle with the viClose function.

4.2.1.2 Realize setting and query functions with VISA library and C language

```
/*************************************************************
This example is used to query and set the sensor frequency, and finally recover the frequency before  setting.
Start VC, and add necessary files. Enter following codes into your .cpp file
*************************************************************/
#include "stdafx.h"
#include <visa.h>
#include <stdio.h>
#include <stdlib.h>

// USB address, “201202006” is the serial number of S8 7 2 3 0. The serial numbers of each S8 7 2 3 0 are different from each other.
void MyDeviceIo(ViSession uiDev);
#define POWER_METER_USB_ADDR            "1204::4100::201803001"
```

```
#define ShowMsg(cMsg)        printf(cMsg)
//Turn on the equipment with the known sensor USB address.
void UsbTest1(void)
{
ViSession uiRm; //Default resource manager handle
ViSession uiDev; //Equipment handle
ViStatus iStatus = VI_SUCCESS; //Status
ViChar rgcDesc[VI_FIND_BUFLEN]; //Equipment descriptor

sprintf(rgcDesc, "USB0::%s::INSTR", POWER_METER_USB_ADDR);
iStatus = viOpenDefaultRM(&uiRm); //Open the default resource manager
if (iStatus)
{
    ShowMsg ("The Resource Manager can't be opened, Please Recheck the Equipment and Connect It\n");
}
else
{
    iStatus = viOpen(uiRm, rgcDesc, VI_NULL, 2000, &uiDev);
    if (iStatus)
    {
        ShowMsg ("The Equipment can't be Turned on, Please Recheck the Equipment and Connect It\n");
    }
    else
    {
        MyDevicelo(uiDev);     //Successfully open a USB device, and start remote control.
        viClose(uiDev);        //Turn off the equipment
    }
    viClose(uiRm);         //Close the default management handle
}
}

//Search for USBTMC equipment for unknown VID, PID, serial numbers of the USB device.
#define MAX_USBTMC_NUM          32        //How many number of USBTMC equipment at most
#define USBTMC_RSRC      "USB[0-9]*::?*INSTR"    //The USBTMC equipment resource matches with the regular
expression
void UsbTest2(void)
{
ViSession uiRm; //Default resource manager handle
ViSession uiDev[MAX_USBTMC_NUM]; //Equipment handle
ViUInt32 uiCnt; //Multiple devices are found
ViFindList findList; //Equipment list
ViStatus iStatus = VI_SUCCESS; //Status
```

```
ViChar rgcDesc[MAX_USBTMC_NUM][VI_FIND_BUFLEN]; //Resource descriptor
ViUInt32 i; //Cyclic variable
iStatus = viOpenUiRm(&uiRm); //Open the default resource manager
if (iStatus)
{
    ShowMsg ("The Resource Manager can't be opened, Please Recheck the Equipment and Connect It\n");
}
else
{
    iStatus = viFindRsrc(uiRm, USBTMC_RSRC, &findList, &uiCnt, rgcDesc[0]);
    for (i=0:i<uiCnt; i++)
    {
        if ((0= =i) || (VI_SUCCESS == viFindNext(findList, rgcDesc[i])))
        {
            if (VI_SUCCESS == viOpen(uiRm, rgcDesc[i], VI_NULL, 2000, &uiDev[i]))
            {
                MyDeviceIo(uiDev[i]);    //Successfully open a USB device, and start remote control.
                viClose(uiDev[i]);     //Turn off the equipment
            }
        }
    }
    viClose(uiRm);     //Close the default management handle
}
}
void MyDeviceIo(ViSession uiDev)
{
ViChar rgcBuf[VI_FIND_BUFLEN]; //Buffer area
ViByte rgbRead[VI_FIND_BUFLEN]; //Read buffer area
ViReal64 rgdFreq[2]; //Frequency
ViUInt32 uiRetCnt; //Read number of bytes
//1) Firstly query the frequency of the sensor, and save it into nFreq[0]
viPrintf(uiDev, "SENS:FREQ?\n");
Sleep(10);
viRead(uiDev, rgbRead, sizeof(rgbRead), &uiRetCnt);
rgbRead[uiRetCnt] = 0; ShowMsg((PCHAR)rgbRead);
sscanf((PCHAR)&rgbRead[0],
"%lf", &rgdFreq[0]);
sprintf(rgcBuf, "The Sensor Frequency is: %lg\n", rgdFreq[0]);
ShowMsg(rgcBuf);
//2) Set the sensor frequency to 16.78 GHz
viPrintf(uiDev, "SENS:FREQ        %lfGHz\n", 16.78);
//3) Query the frequency of the sensor, and save it into nFreq[1]
```

```
viPrintf(uiDev, "SENS:FREQ?\n");
viScanf(uiDev, "%t", rgcBuf);    //Incorporate the query result into the array
ShowMsg(rgcBuf);
sscanf(rgcBuf, "%lf", &rgdFreq[1]);
sprintf(rgcBuf, "The Sensor Frequency is: %lg\n", rgdFreq[1]);
ShowMsg(rgcBuf);
//4) Restore the frequency of the sensor
viPrintf(uiDev, "SENS:FREQ        %lg\n", rgdFreq[0]);
}
```

Chapter 5 Error Description

This chapter will show you how to find out problems and accept after-sales service. and describe error information of the instrument.

- Error Information
- Repair Methods

5.1 Error Information

The instrument adopts two methods to record measurement errors: Error information queue displayed on the front panel and SCPI (remote control mode) error information queue, and these two kinds of error information queue will be saved separately for management.

- Local error information
- Remote control error information

5.1.1 Local Error Information

1) Error information viewing

Operation method through the interface:

If any error information is displayed in the lower right corner of the instrument during use, it indicates that the operation of the software or hardware of the instrument is defective. You can roughly determine type of the fault according to the error code, and take corresponding troubleshooting measures.

The error display area of the instrument can only display one piece error information at one time. As the instrument may have a number of problems at the same time, all error information can be seen by executing the following operations:

Step 1. Press the [System] and then press the [Error List], and the error list window will pop up.

Step 2. The prompt information will be displayed in the window.

Step 3. Use the mouse to browse the error messages and close the dialog.

Step 4. Select the Clear List button to clear the history error information.

2) Description of error messages

If errors are detected during measurement by the instrument, the warning or error information (error abbreviation + detailed description of error) will be displayed on the right side of the status indication area, and the error level will be marked with different colors of the status bar:

Table 5.1 Identification of Error Level of Status Indication Area with Different Colors

Color	Error Level	Error Description
Red	Severe error	It means the severe error during the measurement. When such error occurs, the instrument can't normally work.
Red	Error	It means the error during the measurement, for example, failure to complete the measurement normally due to data loss or incorrect setting.
Black	Normal	No error information is displayed.

5.1.2 Remote Control Error Information

1) Format and description of error information

Under the remote control mode, the error information will be recorded in the error/event queue of the status reporting system. The error information can be queried with the command "SYSTem:ERRor?", and the format is as follows:

"<Error Code>, "<Error Information in the Error Queue>; <Detailed Description of Error Information>"

Example:

"-110, "Data out of Range; Inputted Parameter out of Lower Bound."

The remote control error information includes two types:

- The information of the negative error code defined in SCPI standard will not be described in details here.
- The specific description of the instrument characteristics positive error code is as shown in the following table:

Table 5.2 Description of the Instrument Characteristics Error Information

Error Code	Error Description
-101	Invalid character Invalid character: There is invalid character in the program mnemonic (command or parameter). For example: AVER:COUN !6
-102	Syntax error Syntax error: The program mnemonic syntax is invalid. For example: DISPlay:ACT, CH1
-108	Parameter not allowed Parameter not allowed: The command has too many parameters, or no parameter follows the parameter command. For example: TRAC:AUT ON
-109	Missing parameter Missing parameter: The command has too few parameters. For example: AVER
-112	Program mnemonic too long Program mnemonic too long: A single segment in a command has more than 12 characters. For example: OUTPutROSCillatorSTATe ON
-113	Undefined header Undefined header: The instrument receives an unrecognized command. Possible reasons: Incorrect spelling or incorrect abbreviation of the command. For example: CALI:AUTO
-121	Invalid character in number Invalid character in number: There is an invalid character in the numeric data. For example: SENS:CORR:GAIN2 #12
-123	Exponent too large Exponent too large: The exponent of the numeric data is larger than 32,000. For example: SENS:CORR:GAIN2 1E32001
-124	Too many digits Too many digits: The magnitude of the numeric data is more than 255 bits, excluding the prefixed 0.
-128	Numeric data not allowed Numeric data not allowed: The command which can't receive the numeric data receives a value.
-131	Invalid suffix Invalid suffix: The suffix of the numeric data is incorrect. For example: FREQ 10GZ
-134	Suffix too long Suffix too long: The suffix is longer than 12 characters. For example: FREQ 10GHHHHHHHHHHHHHHHHHZ

-138	Suffix not allowed Suffix not allowed: The numeric data can't be followed by a suffix. For example: SENS:CORR:GAIN2 12HZ
-148	Character data not allowed Character data not allowed: Check if a quotation mark shall be added. For example: MEM:CLE State_1 Correct: MEM:CLE "State_1"
-151	Invalid string data Invalid string data: Check if the single or double quotation marks of the string are matched. For example: MEM:CLE "State1
-158	String data not allowed String data not allowed: Check if the parameter type is valid. For example: OUTP:ROSC "ON"
-161	Invalid block data Invalid block data: Check it according to section 7.7.6 of the IEEE 488.2.
-168	Block data not allowed Block data not allowed: The valid data block is detected, but it is not supported by the command. For example: OUTP:ROSC #15FETC?
-178	Expression data not allowed Expression data not allowed: The valid expression is detected, but it is not allowed in the instrument. For example: SENS:CORR:GAIN2 (1+3)
-211	Trigger ignored Trigger ignored: The TRIG:IMM, *TRG commands are received when the instrument is not in the waiting for trigger status.
-213	Init ignored Init ignored: The measurement initialization command is received when the instrument has been initialized. For example: INIT:CONT ON INIT
-214	Trigger deadlock Trigger deadlock.
-220	Parameter error; Frequency list must be in ascending order. Parameter error, frequency list must be in ascending order.
-221	Settings conflict Settings conflict: Many causes will lead to a conflict, such as setting trigger delay during statistical measurement.
-222	Data out of range Data out of range: The numeric data is not within the valid range. For example: AVER:COUN 100000
-224	Illegal parameter value Illegal parameter value: A discrete parameter is received, but it is invalid for this command.
-226	Lists not same length Lists not same length.
-230	Data corrupt or stale; Please calibrate Data corrupt or stale; Please calibrate

-231	Data questionable; CAL ERROR Data questionable; CAL ERROR: The calibration of the instrument fails. The most possible cause is that no sensor is connected to the output end of the calibration source during calibration. Data questionable; CAL ERROR Ch1 Data questionable; CAL ERROR: The calibration of the instrument fails. The most possible cause is that no sensor is connected to the output end of the calibration source during calibration. Data questionable; Input Overload Data questionable; Input Overload: The power input exceeds the upper power limit of the sensor. Data questionable; Input Overload Ch1 Data questionable; Input Overload: The power input exceeds the upper power limit of the sensor. Data questionable; Lower window log error Data questionable; Lower window log error: In case of the difference measurement, the measurement result is 0 and the unit displayed is logarithm. Data questionable; Upper window log error Data questionable; Upper window log error: In case of the difference measurement, the measurement result is 0 and the unit displayed is logarithm. Data questionable; ZERO ERROR Data questionable; ZERO ERROR: The zero of the instrument fails. The most possible cause is that the power signal is inputted during zero. Data questionable; ZERO ERROR Ch1 Data questionable; ZERO ERROR: The zero of the instrument fails. The most possible cause is that the power signal is inputted during zero.
-241	Hardware missing Hardware missing: The instrument fails to execute the command, and the cause is that no sensor is connected or the sensor type is unmatched.
-310	System error; Sensor EEPROM Read Failed - critical data not found or unreadable System error; Sensor EEPROM Read Failed - critical data not found or unreadable System error; Sensor EEPROM Read Failed - unknown EEPROM table format System error; Sensor EEPROM Read Failed - unknown EEPROM table format.
-321	Out of memory Out of memory.
-330	Self-test Failed; Self-test failed.
-350	Queue overflow Queue overflow: After the error queue is full, the subsequent errors will not be recorded.
-410	Query INTERRUPTED Query INTERRUPTED: A command shall send data to the output buffer area, but the buffer area has already included the data sent by the previous command (the previous data will not be overwritten). The output buffer area will be cleared when the machine is powered off or when *RST command is received. For details, please refer to section 6.3.2.3 of IEEE488.2.
-420	Query UNTERMINATED Query UNTERMINATED: The instrument is set to speak (namely, send data to the interface bus), but no command to send data to the output buffer area is received. For details, please refer to section 6.3.2.2 of IEEE488.2. For example, try to read data from a remote interface after executing CONFigure command (it will not generate data).
-430	Query DEADLOCKED Query DEADLOCKED: The output buffer area can't contain the command to generate too much data, and the output buffer area is full. The command continues but data is lost. For details, please refer to section 6.3.1.7 of IEEE488.2.

-440	Query UNTERMINATED after indefinite response Query UNTERMINATED after indefinite response: Some combined queries may generate illegal response information. If the query command of an indefinite response (any block response with indefinite length or any ASCII response data) is not the last query command, the instrument will report the query error, and it will send the response any more after the query command. For details, please refer to section 6.5.7.5 of IEEE488.2.

2) Error information type

Error events correspond to only one type of error message. The types of error information are classified and described below:

- **Query error (-499 to -400):** It means that the message exchange protocol error described in IEEE 488.2, Chapter 2 is detected by the output queue control of the instrument. In this case, the query error bit (bit2) of the event status register will be set (for details, see IEEE 488.2, 6.5). In this case, the data can't be successfully read from the output queue.
- **Instrument characteristics error (-399 to -300, 201 to 703, and 800 to 810):** It means that the instrument operation fails, which may be caused by abnormal hardware or firmware status. Such error code is often used for the instrument self-test. In this case, the instrument characteristics error bit (bit3) of the event status register will be set.
- **Execution error (-299 to -200):** It means that errors are detected during measurement. In this case, the execution error bit (bit4) of the event status register will be set.
- **Command error (-199 to -100):** It means the syntax error detected during the instrument command parse, which is caused by wrong command format generally. In this case, the command error bit (bit5) of the event status register will be set.

5.2 Repair Method

- Contact Us
- Packing and Delivery

5.2.1 Contact Us

If S87230 series USB CW power sensor has any fault, firstly observe and save the error information, and then analyze possible causes and refer to methods described in section "5 Fault Diagnosis and Troubleshooting" of user manual to do troubleshooting in advance. If the fault is not solved, please contact our service center according to the following contact information and provide the collected error information, and we will help you solve the problem as soon as possible.

Contact information:

Service Tel: 886.909 602 109

Website: www.salukitec.com

Email: sales@salukitec.com

Address: No. 367 Fuxing N Road, Taipei 105, Taiwan (R.O.C.)

5.2.2 Packing and Delivery

When your instrument incurs a problem that is difficult to solve, please contact us by phone or fax. If it is decided that the instrument needs to be returned for repair, please pack it with the original packaging material and box as per the following

steps:

1) Please include a detailed explanation of the problem that you've encountered when using the instrument along with the apparatus in the packaging box.
2) Pack the instrument with the original packaging material to reduce possible damage.
3) Place the pads at the corners of outer packing box and then put the instrument into the packing box.
4) Seal the packing box with tapes, and reinforce it with nylon tape;
5) Mark “Fragile! No Touch! Handle with Care!” indicated.
6) Please arrange the consignment as required for the precise instrument.
7) Keep copies of all the shipping documents.

Note

Pay attention to followings when packaging the instrument

Packaging the instrument with other materials may damage the instrument. It is forbidden to use polystyrene beads as the packaging material because they can't fully protect the instrument and may damage the instrument after being sucked into the instrument fan by the static electricity.

Appendixes

Appendix A Lookup Table of the SCPI by Subsystem

Table 6.1 Lookup Table of the SCPI by Subsystem

Command	Operation	Brief introduction to functions
*CLS	Setting only	Clear the instrument status data structure
*ESE		Query or set the standard event status enable register
*ESR?	Query only	Query the value of the standard event status register
*IDN?	Query only	Query identification string of the instrument
*OPC		When all pending operations are completed, set the operation end bit in the standard event status register
*RCL	Setting only	Recall the status in the specified save recall register
*RST	Setting only	Reset the instrument
*SAV	Setting only	Save the instrument status in the specified register
*SRE		Query or set the service request register
*STB?	Query only	Query the status word
*TRG	Setting only	Trigger all channels to be triggered
*TST?	Query only	Execute self-test
*WAI	Setting only	Keep the instrument in a waiting status
:ABORt[1]	Setting only	Disable measurement of the instrument
:CALCulate[1]:LIMit:CLEar:AUTO		Control the time to reset the limit FCO (failure count)
:CALCulate[1]:LIMit:CLEar[:IMMediate]	Setting only	Reset FCO (failure count)
:CALCulate[1]:LIMit:FAIL?	Query only	Query if it is out of the limit
:CALCulate[1]:LIMit:FCOunt?	Query only	Query the limit detection failure count (FCO)
:CALCulate[1]:LIMit:LOWer[:DATA]		Query or set the lower measurement limit
:CALCulate[1]:LIMit:STATe		Query or set the measurement limit detection switch
:CALCulate[1]:LIMit:UPPer[:DATA]		Query or set the upper measurement limit
:CALCulate[1]:MATH[:EXPRession]		Query or set the specified measurement expression
:CALCulate[1]:MATH[:EXPRession]:CATalogue?	Query only	List all measurement expressions
:CALCulate[1]:RELative[:MAGNitude]:AUTO		Set the reference value for relative measurement
:CALCulate[1]:RELative[:MAGNitude]:VALue?	Query only	Query the reference value for relative measurement
:CALCulate[1]:RELative:STATe		Query or set the relative measurement switch status

:CALibration[1]:ZERO:AUTO		Zero the instrument
:CALibration[1]:ZERO:TYPE		Query or set the zero type of the sensor
:CONFigure[1][:SCALar][:POWer][:AC]		Query or set the power measurement mode
:CONFigure[1][:SCALar][:POWer][:AC]:RELative	Setting only	Set the absolute power measurement mode, and enable relative measurement
:FETCh[1][:SCALar][:POWer][:AC]?	Query only	Set the absolute power measurement, disable relative measurement, and return the measurement value
:FETCh[1][:SCALar][:POWer][:AC]:RELative?	Query only	Set the absolute power measurement, enable relative measurement, and return the measurement value
:FORMat[:READings]:BORDer		Query or set the transmission order of binary data
		Query or set the data transmission format
:INITiate[1]:CONTinuous		Query or set the trigger status
:INITiate:CONTinuous:ALL		Query or set the trigger status of all the channels
:INITiate:CONTinuous:SEQuence[1]		Query or set the trigger status
:INITiate[1][:IMMediate]	Setting only	Set the instrument to the waiting for trigger status
:INITiate[:IMMediate]:ALL	Setting only	Set all the channels to the waiting for trigger status
:INITiate[:IMMediate]:SEQuence[1]	Setting only	Set the instrument to the waiting for trigger status
:MEASure[1][:SCALar][:POWer][:AC]?	Query only	Set the absolute power measurement, disable relative measurement, and return the measurement value
:MEASure[1][:SCALar][:POWer][:AC]:RELative?	Query only	Set the absolute power measurement, enable relative measurement, and return the measurement value
:MEMory:CATalog[:ALL]?	Query only	List the user configuration in the instrument, including save recall configuration and FDO table
:MEMory:CATalog:STATe?	Query only	List the save recall configuration in the instrument
:MEMory:CATalog:TABLe?	Query only	List FDO table in the instrument
:MEMory:CLEar[:NAME]	Setting only	It is used to reset the specified FDO table or save recall table
:MEMory:CLEar:TABLe[1]	Setting only	Clear the specified FDO table
:MEMory:FREE[:ALL]?	Query only	Query the total number of unused bytes and the number of used bytes in the user configuration space
:MEMory:FREE:STATe?	Query only	Query the total number of unused bytes and the number of used bytes in the save recall space
:MEMory:FREE:TABLe?	Query only	Query the total number of unused bytes and the number of used bytes in FDO table space
:MEMory:NSTates?	Query only	Query number of the save recall status, and 10 will be returned always
:MEMory:STATe:CATalog?	Query only	List name of all the save recall status
:MEMory:TABLe[1]:DEFine		Query or set name of the specified FDO table
:MEMory:TABLe[1]:FREQuency		Query or set the frequency list of the specified FDO table

:MEMory:TABLe[1]:FREQuency:POINts?	Query only	Query number of the frequency point of the specified FDO table
:MEMory:TABLe[1]:GAIN[:MAGNitude]		Query or set the amplitude gain list of the specified FDO table
:MEMory:TABLe[1]:GAIN[:MAGNitude]:POINts?	Query only	Query number of the amplitude gain point of the specified FDO table
:MEMory:TABLe:MOVE	Setting only	Rename the specified FDO table
:MEMory:TABLe:SELect		Query or set the current FDO table
:READ[1][:SCALar][:POWer][:AC]?	Query only	Set the absolute power measurement, disable relative measurement, and return the measurement value
:READ[1][:SCALar][:POWer][:AC]:RELative?	Query only	Set the absolute power measurement, enable relative measurement, and return the measurement value
[:SENSe[1]:AVERage[1]:COUNt		Query or set the average number of times of the channel
[:SENSe[1]:AVERage:COUNt:AUTO		Query or set the automatic average status of the channel
[:SENSe[1]:AVERage:SDETect		Query or set the step detection status of the channel
[:SENSe[1]:AVERage[1][:STATe]		Query or set the average switch status of the channel
[:SENSe[1]:CORRection:CSET2[:SELect]		Select FDO table
[:SENSe[1]:CORRection:CSET:STATe		Query or set the enable status of FDO table
[:SENSe[1]:CORRection:FDOFfset|GAIN4[:INPut][:MAGNitude]?	Query only	Query the frequency response offset factor
[:SENSe[1]:CORRection:GAIN2:[INPut][:MAGNitude]		Query or set the offset of the channel
[:SENSe[1]:CORRection:GAIN2:[INPut]:STATe		Query or set the offset enable status of the channel
[:SENSe[1]:FREQuency[:CW|FIXed]		Query or set the frequency of the channel
[:SENSe[1]:POWer:AC:RANGe		Query or set range of the sensor
[:SENSe[1]:POWer:AC:RANGe:AUTO		Query or set the automatic range switch status of the sensor
:SERVice:LANGuage		Query or set the language selection
:SERVice:SENSor[1]:CDATe?	Query only	Query the sensor calibration date
:SERVice:SENSor[1]:CPLace?	Query only	Query the sensor calibration location
:SERVice:SENSor[1]:FREQuency:MAXimum?	Query only	Query the maximum frequency
:SERVice:SENSor[1]:FREQuency:MINimum?	Query only	Query the minimum frequency

:SERVice:SENSor[1]:SNUMber?	Query only	Query the serial number
:SERVice:SENSor[1]:TYPE?	Query only	Query the type
:SERVice:SNUMber		Query or set the serial number of the instrument
:STATus:DEVice:CONDition?	Query only	Query the value in the equipment status condition register
:STATus:DEVice:ENABle		Query or set the equipment status event enable register
:STATus:DEVice[:EVENt]?	Query only	Query the equipment event register
:STATus:DEVice:NTRansition		Query or set the equipment negative jump filter
:STATus:DEVice:PTRansition		Query or set the equipment positive jump filter
:STATus:OPERation:CALibrating[:SUMMary]:CONDition?	Query only	Query the value in the calibration status condition register
:STATus:OPERation:CALibrating[:SUMMary]:ENABle		Query or set the calibration operation event enable register
:STATus:OPERation:CALibrating[:SUMMary][:EVENt]?	Query only	Query the calibration operation event register
:STATus:OPERation:CALibrating[:SUMMary]:NTRansition		Query or set the calibration negative jump filter
:STATus:OPERation:CALibrating[:SUMMary]:PTRansition		Query or set the calibration positive jump filter
:STATus:OPERation:CONDition?	Query only	Query the value in the operation status condition register
:STATus:OPERation:ENAB le		Query or set the operation status event enable register
:STATus:OPERation[:EVENt]?	Query only	Query the operation status event register
:STATus:OPERation:LLFail[:SUMMary]:CONDition?	Query only	Query the value in the lower limit detectio operation status condition register
:STATus:OPERation:LLFail[:SUMMary]:ENABle		Query or set the lower limit detection operation event enable register
:STATus:OPERation:LLFail[:SUMMary][:EVENt]?	Query only	Query the lower limit detection operation event register
:STATus:OPERation:LLFail[:SUMMary]:NTRansition		Query or set the lower detection operation negative jump filter
:STATus:OPERation:LLFail[:SUMMary]:PTRansition		Query or set the lower detection operation positive jump filter
:STATus:OPERation:NTRansition		Query or set the operation status negative jump filter
:STATus:OPERation:PTRansition		Query or set the operation status positive jump filter
:STATus:OPERation:SENSe[:SUMMary]:CONDition?	Query only	Query the value in the sense operation status condition register

:STATus:OPERation:SENSe[:SUMMary]:ENABle		Query or set the sense operation event enable register
:STATus:OPERation:SENSe[:SUMMary][:EVENt]?	Query only	Query the sense operation event register
:STATus:OPERation:SENSe[:SUMMary]:NTRansition		Query or set the sense operation negative jump filter
:STATus:OPERation:SENSe[:SUMMary]:PTRansition		Query or set the sense operation positive jump filter
:STATus:OPERation:ULFail[:SUMMary]:CONDition?	Query only	Query the value in the upper limit detection operation status condition register
:STATus:OPERation:ULFail[:SUMMary]:ENABle		Query or set the upper limit detection operation event enable register
:STATus:OPERation:ULFail[:SUMMary][:EVENt]?	Query only	Query the upper limit detection operation event register
:STATus:OPERation:ULFail[:SUMMary]:NTRansition		Query or set the upper detection operation negative jump filter
:STATus:OPERation:ULFail[:SUMMary]:PTRansition		Query or set the upper detection operation positive jump filter
:STATus:PRESet	Setting only	Preset some status registers
:STATus:QUEStionable:CALibration[:SUMMary]:CONDition?	Query only	Query the value in the calibration questionable status condition register
:STATus:QUEStionable:CALibration[:SUMMary]:ENABle		Query or set the calibration questionable event enable register
:STATus:QUEStionable:CALibration[:SUMMary][:EVENt]?	Query only	Query the calibration operation event register
:STATus:QUEStionable:CALibration[:SUMMary]:NTRansition		Query or set the calibration questionable negative jump filter
:STATus:QUEStionable:CALibration[:SUMMary]:PTRansition		Query or set the calibration questionable positive jump filter
:STATus:QUEStionable:CONDition?	Query only	Query the value in the questionable status condition register
:STATus:QUEStionable:ENABle		Query or set the questionable status event enable register
:STATus:QUEStionable[:EVENt]?	Query only	Query the questionable status event register
:STATus:QUEStionable:NTRansition		Query or set the questionable status negative jump filter
:STATus:QUEStionable:POWer[:SUMMary]:CONDition?	Query only	Query the value in the power questionable status condition register
:STATus:QUEStionable:POWer[:SUMMary]:ENABle		Query or set the power questionable event enable register

:STATus:QUEStionable:POWer[:SUMMary][:EVENt]?	Query only	Query the power questionable event register
:STATus:QUEStionable:POWer[:SUMMary]:NTRansition		Query or set the power questionable negative jump filter
:STATus:QUEStionable:POWer[:SUMMary]:PTRansition		Query or set the power questionable positive jump filter
:STATus:QUEStionable:PTRansition		Query or set the questionable status positive jump filter
:SYSTem:ERRor[:NEXT]?	Query only	Return the error code and error information of the instrument from its error queue
:SYSTem:HELP:HEADers?	Query only	Query the command list supported by the instrument
:SYSTem:PRESet	Setting only	Reset the instrument to the status specified by the parameter
:SYSTem:VERSion?	Query only	Query SCPI version number of the instrument
:TRIGger[1][:IMMediate]	Setting only	Set the instrument to the waiting for trigger status
:UNIT[1]:POWer		Query or set the power unit
:UNIT[1]:POWer:RATio		Query or set the ratio measurement power unit