87237 Series USB Thermal Power Sensor Programmed Control Manual



Ceyear Technologies Co., Ltd.

This manual applies to the following models of USB Thermal Power Sensor based on the firmware version of 1.0 and higher.

- 87237D USB Thermal Power Sensor, frequency range DC~18GHz, power range -35dBm~+20dBm
- 87237L USB Thermal Power Sensor, frequency range DC~67GHz, power range -35dBm~+20dBm
- 87237P USB Thermal Power Sensor, frequency range DC~110GHz, power range -30dBm~+20dBm

Version: A.1 February 2026, Ceyear Technologies Co., Ltd.
Address: No. 98, Xiangjiang Road, Huangdao District, Qingdao, Shandong Province
Service consultation: 400-1684191
Quality supervision: 0532-86886614
Website: www.ceyear.com
Email: techbb@ceyear.com
Postal code: 266555

Foreword

Thank you for choosing and using the 87237 series USB Thermal power sensor developed and produced by Ceyear Technologies Co., Ltd.! This product features quality, precision and excellence, and has a high cost performance among similar products.

We will take the responsibility to maximally meet your needs and provide you with high-quality measuring instruments and first-class after-sales service. We aim to provide "high quality and considerate service", and operate on the principle of making customers satisfactory with our products and services.

Manual No.

YQ2.984.1710SCCN

Version

A.1 2026.02

Ceyear Technologies Co., Ltd.

Manual authorization

The contents of this manual are subject to change without notice. The contents and terms used in this manual are interpreted by Ceyear Technologies Co., Ltd.

The copyright of this manual belongs to Ceyear Technologies Co., Ltd, no modification can be made to this manual contents by any unit or person without approval of this institute, and no reproduction or propagation of this manual can be made for profits. Otherwise, Ceyear Technologies Co., Ltd reserves the right of pursuing legal responsibilities from any infringer.

Product warranty

The warranty period of this product is 18 months from the date of shipment. The instrument manufacturer will repair or replace the damaged components according to the user's requirements and actual conditions within the warranty period. For specific maintenance issues, refer to the contract.

Product quality certificate

This product is guaranteed to meet the specifications in this manual from the date of shipment. The calibration and measurement are completed by measuring bodies with national qualification, with relevant data to be provided for reference by users.

Quality/environmental management

This product complies with the quality and environmental management systems during R&D, manufacturing and testing. Ceyear Technologies Co., Ltd. is qualified and has passed ISO 9001 and ISO 14001 management systems.

Safety precautions

NOTICE

Notice mark, indicating some important information which will not cause danger. It reminds the user to pay attention to a certain operation process, operation method or the like. Failure to observe the rules or operate correctly may cause damage to the instrument or loss of important data. Proceed to the next step only after fully understanding and meeting the notice conditions indicated.

TIPS

The "Tips" symbol indicates information tips. It reminds the user to pay attention to the instrument or certain operation process, operation method or the like. The purpose is to guide the instrument operator to use the instrument correctly.

Contents

1. Manual navigation····· 1

1.1. About this manual·····1
1.2 Related documents·····2

2. Remote control····· 4

2.1 Remote control basics····· 4
2.1.1 Programmed control interface·····4
2.1.2 Message····· 9
2.1.3 SCPI····· 10
2.1.4 Command sequence and synchronization····· 21
2.1.5 State reporting system····· 23
2.1.6 Precautions for programming·····32
2.2 Instrument programmed control port and configuration····· 33
2.3 I/O library····· 33
2.3.1 Overview of I/O library····· 33
2.3.2 Installation and configuration of I/O library····· 35
2.4. Zeroing····· 35
2.5 Performing measurements····· 36
2.6 Using frequency response offset table····· 36
2.6.1 Overview·····36
2.6.2 Entering frequency offset table·····37
2.6.3 Selecting frequency offset table·····37
2.6.4 Enabling frequency offset table····· 38
2.6. 5 Measurement applications·····38
2.6.6 Specific Applications·····38
2.7 Setting Display Resolution····· 38
2.8 Setting average····· 39
2.9 Setting Offset····· 39
2.10 Setting measurement limits····· 39
2.11 State report····· 40
2.12 Storage call····· 41

3. Programmed control commands····· 43

3.1 Description of commands·····43
3.2 General commands(IEEE488.2 Commands)····· 43
*CLS·····44
*DDT·····44
*ESE·····45
*ESR?····· 46
*IDN?····· 46
*OPC·····46
*ESE?····· 47
*RCL····· 47
*RST·····47
*SAV····· 47
*SRE·····48
*STB?····· 48
*TRG····· 49
*TST?····· 49
*WAI····· 49
3.3 Instrument subsystem command·····50
3.3.1 Calculation subsystem command (CALCulate)····· 50
3.3.2Calibration sub-system command (CALibration)·····59
3.3.3 Measurement subsystem command (FETCh/READ/MEASure)·····61
3.3.4 Display subsystem command (DISPlay)····· 71
3.3.5 Format subsystem command (FORMat)····· 71
3.3.6 Memory subsystem command (MEMory/MMEMory)····· 72
3.3.7 Sensor subsystem command (SENSe)····· 80
3.3.8 State subsystem command·····92
3.3.9 System subsystem command (SYSTem)·····138
3.3.10 Trigger subsystem command (INITiate/ TRIGger)····· 142
3.3.11 Unit subsystem command (UNIT)····· 148
3.3.12 Service subsystem command (SERVice)·····149

4. Programming examples····153

4.1 Examples of basic operation···· 153
4.1.1 VISA library···· 153
4.1.2 Examples of runtime environment···· 154
4.1.3 Initialization and setting default state···· 155
4.1.4 Sending setting command···· 156
4.1.5 Read the state of measuring instrument···· 157
4.1.6 Command synchronization···· 157
4.2 Examples of advanced operation···· 158
4.2.1 Examples of USBTMC programmed control···· 158
4.2.2 Linux development examples···· 161
4.3 Application Examples···· 163
4.3.1 External trigger buffer measurement···· 163
4.3.2 External trigger buffer sweep frequency measurement···· 163
4.3.3 Fast measurement mode (free run)···· 163
4.3.4 Fast measurement mode (external trigger)···· 164
4.3.5 Continuous and uninterrupted measurements···· 164

5. Error description···· 167

5.1 Error message···· 167
1) Error message format and description···· 167
2) Error message type···· 171
5.2 Method to obtain after-sales services···· 172
5.2.1 Contacting us···· 172
5.2.2 Package and mailing···· 172

Annex···· 175

Annex A Zoom table of SCPI classified by subsystem···· 175

1. Manual navigation

This chapter introduces the programmed control manual functions, chapter structure and main contents of the 87237 series USB Thermal power sensor (hereinafter to as the 87237), as well as the instrument-related documents provided to users.

- About this manual.....1
- Related Documents.....2

1.1. About this manual

This manual introduces the methods for remote control of the 87237 and application of SCPI. Meanwhile, in order to make it convenient for users to quickly master the remote control programming methods, some programming examples are listed, and the basic concept of I/O function library is introduced. To facilitate your skillful use of such instrument, please read carefully and follow this manual in advance for correct operation.

SCPI (standard commands for programmable instruments) define the standards and methods for remote control of the instrument, and are the programmed control programming language for programmed control electronic test and measuring instruments. SCPI are based on the IEEE 488.2 standard and form. Please access http://www.scpiconsortium.org.

The manual details the programmed control command of the 87237.

The chapters of the programmed control manual include:

- Remote control

  The methods for remote control of the instrument are summarized to make users get familiar with remote control quickly. It is divided into three parts: remote control basics, introducing program related concepts, software configuration, program port, SCPI, etc.; instrument port configuration method, introducing the connection method and software configuration method for program ports of the 87237; I/O function library, introducing the basic concept of instrument driver and basic installation instructions of IVI-COM/IVI-C driver.

- Programmed control commands

  Common commands, instrument commands and compatible commands are introduced, and the functions, parameters and examples of SCPI are described.

- Programming examples

  The basic programming examples and advanced programming examples are provided in the way of text description and example code, and the explanation is provided to make it convenient for users to quickly master the remote control

programming method of the 87237.

- Error description

  Error message description and method to obtain after-sales services are included.

- Annex

  Necessary reference information related to programmed control of the 87237 is provided, including zoom table of SCPIs.

1.2 Related documents

The product documentation includes

- Quick start guide
- User's manual
- Programmed control manual

Quick start guide

This manual introduces the basic methods for configuration and start-up measurement of the instrument to enable users to quickly understand the characteristics of the instrument, and master the basic setting and operation methods. Main chapters include:

- Get prepared
- Typical applications
- Get help

User's manual

This manual describes the functions and operation methods of the analyzer in detail, including configuration, measurement, programmed control and maintenance, etc. The purpose is to guide the user to fully understand the functional characteristics of the product and master common test methods of the instrument. Main chapters include:

- Manual navigation
- Overview
- Quick start
- Operation guide
- Menus
- Remote control
- Troubleshooting and repair
- Technical indicators and test methods
- Annex

Programmed control manual

This manual introduces remote programming basics, SCPI basics, SCPI, programming examples and I/O driver function library in detail. The purpose is to guide

users to quickly and comprehensively master the programmed control commands and methods of the instrument. Main chapters include:

- Remote control
- Programmed control commands
- Programming examples
- Error description
- Annex

2. Remote control

This chapter introduces the remote control basics, remote control interface and configuration methods of the 87237, and briefly introduces the concept and classification of I/O instrument driver library. The purpose is to facilitate users to start to achieve remote control. Specific contents include:

- Remote control basics....4
- Instrument programmed control port and configuration....33
- I/O library....35
- Zeroing....35
- Performing measurements....36
- Using frequency response offset table....36
- Setting display resolution....38
- Setting average....39
- Setting offset....39
- Setting measurement limit....39
- State report....40
- Storage call....41

2.1 Remote control basics

- Programmed control interface....4
- Message....9
- SCPI command....10
- Command sequence and synchronization....21
- State reporting system....23
- Precautions for programming....32

2.1.1 Programmed control interface

- LAN Interface....5
- GPIB Interface....7
- RS-232 Interface....8
- USB Interface....8

Instruments with remote control function generally support two kinds of remote control interface: LAN, GPIB, RS-232 and USB, and the type of port supported by the specific model of instrument is determined by the function of the instrument.

The programmed control interface and related VISA addressing string are described in the table below:

Table 2.1 Remote control interface type and VISA addressing string

Programmed control interface	VISA addressing string	Description
LAN (local area network)	VXI-11 protocol: TC PIP::Addressograph[::LAN_device_name][::INSTR] Raw socket protocol: TC PIP::Addressograph::port::SOCKET	The controller realizes remote control by connecting the instrument with the network port on the rear panel of the instrument. For details of the protocol, please refer to: 2.1.1.1 LAN interface
GPIB (IEC/IEEE Bus Interface)	GPIB::primary address[::INSTR]	The controller realizes remote control by connecting the instrument with the port on the rear panel of the instrument. The bus interface standard IEC 625.1/IEEE 418 is met. For details, please refer to: 2.1.1.2 GPIB interface
RS-232 (RecommendedStandard-232)		Instrument 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 rear panel port. For details, please refer to: 2.1.1.4 USB interface

2.1.1.1 LAN interface

An instrument with a network interface (hereinafter referred to as instrument) can be controlled remotely by computers in 10Base-T and 100Base-T. Various instruments are combined into a system in LAN and controlled uniformly by computers in it. In order to realize remote control in LAN, it shall be equipped with port connector, network card and relevant network protocol in advance, and provided with relevant network services. Meanwhile, the host computer in the network shall also be equipped with instrument control software and VISA library in advance. The three operation modes of the network card are:

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

The controller computer and the instrument shall be connected to the common TCP/IP protocol network through the network port. The cable between the computer

and the instrument is a commercial RJ45 cable (Category 5 cable with or without shielding). During data transmission, the transmission speed of LAN is faster when data packet transmission is applied. Generally, the length of the cable between the computer and the instrument shall not exceed 100m (100Base-T and 10Base-T). For more information about LAN communications, please refer to http://www.ieee.org.http://www.ieee.org/
Knowledges about the LAN interface are introduced below:

1) IP address

Physical connection of the network shall be guaranteed for remote control on the instrument via the LAN. It is just required to set the address to the subnet in which the host computer is located via the network configuration interface of the instrument. For example, if the IP address of the host computer is 192.168.12.0, the IP of the instrument shall be set to 192.168.12.XXX, whereas XXX is the figure between 1 and 255.
When establishing a network connection, only the IP address is required. The VISA addressing string is as follows:

TC PIP::host address[::LAN device name][::INSTR] or
TCPIP:: host address: port:: SOCKET
Where:

- TCPIP represents the network protocol used;
- The host address represents the IP address or name of the instrument, and is used for identifying and controlling the controlled instrument;
- LAN device name defines the handle number of the protocol and subset (optional);
  - VXI-11 protocol is selected for device 0;
  - More recent high speed LAN instrument protocol is selected for high speed LAN instrument 0;
- INSTR represents the instrument resource type (optional);
- port represents the socket port number;
- SOCKET represents the raw network socket resource class.

Example:

- The IP address of the instrument is 192.1.2.3, and the effective resource string of the VXI-11 protocol is:
  TCPIP::192.1.2.3::INSTR
- To establish a raw socket connection, use:
  TCPIP::192.1.2.3::5025::SOCKET

TIPS

Method of recognizing multiple instruments in the programmed control system

If multiple instruments are connected in the network, the individual IP address and related resource string are used to distinguish. The host computer applies its own 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 associated network services are pre-configured for communication. Such connection-oriented communication, which follows the sequential exchange and can identify the interruption of the connection, ensures no loss of information.

3) Socket communication

The TCP/IP protocol connects the instrument to the network via LAN sockets. As a basic method used in computer network programming, the socket allows applications using different hardware and operation systems to communicate over a network. With this method, two-way communication between the instrument and the computer is realized through ports.

As a software class programmed specially, the socket defines the IP address, device port number and other necessary information for network communication, and integrates some basic operations in network programming. Sockets can be used after installing packaged libraries in the operation system. Two commonly used socket libraries are the Berkeley socket library for UNIX the Winsock library for Windows.

Sockets in the instrument are compatible with Berkeley sockets and Winsock through the application program interface (API). In addition, it is compatible with the API of other standard sockets. When SCPI are used to control the instrument, the socket program established in the program issues the command. Before using a LAN socket, the socket port number of the instrument must be set. The socket port number of the instrument is 5025.

2.1.1.2 GPIB interface

As an instrument programmed control interface widely used at present, GPIB interface is connected to different types of instruments through GPIB cable, so as to build a test system with the host computer. In order to realize remote control, the host computer shall be equipped with GPIB bus card, driver and VISA library in advance.

2.1 Remote control basics

During communication, the host computer first addresses the controlled instrument through the GPIB bus address. The user may set the GPIB address and ID query string, and the GPIB communication language may be in the form of SCPI by default.
GPIB and its associated interface operations are defined and described in detail in ANSI/IEEE Standard 488.1-2003 and ANSI/IEEE Standard IEEE Standard 488.2-1992. For details of the standards, please refer to the IEEE website: http://www.ieee.org/http://www.ieee.org.

GPIB processes information in bytes at the data transmission speed of up to 8MBps, which is fast. Since the data transmission rate is limited by the distance between the device/system and the computer, the following points shall be noted when connecting GPIB:

- Up to 15 instruments may be built through GPIB interface.
- The total length of the transmission cable shall not be more than 15 m or twice the number of instruments in the system. In general, the maximum length of the transmission cable between the devices shall not exceed 2 m;
- If multiple instruments are connected in parallel, a "live" cable is required;
- The end of the IEC bus cable shall be connected to the instrument or controller computer.

2.1.1.3 RS-232 interface

RS-232 is a traditional method to realize programmed control. Because it only sends and receives one bit of data at a time, the transmission rate is slower than GPIB or LAN, which is not commonly used at present. Similar to GPIB and LAN, instrument parameters, such as baud rate, need to be set when establishing communication in order to match the parameters with the host computer. RS-232 transmits SCPI command characters in ASCII form.

2.1.1.4 USB interface

To implement USB programmed control, a computer and signal generator need to be connected via a USB port with the VISA library installed in advance. VISA automatically detects and configures the instrument to establish a USB connection without the need to enter the instrument address string or install a separate driver.

USB address:

Addressing string format: USB::<vendor ID>::<product ID>::<serial number>[::RAW]

Where:

- <vendor ID> represents the manufacturer code;
- <product ID> represents the instrument code;
- <serial number> represents the serial number of the network analyzer;

Example:

USB0::0x3399::0x3830::2202003::0::INSTR
0x3399: Manufacturer code;
0x3830: Instrument code;
2202003: serial number of the network analyzer.

2.1.2 Message

The messages transmitted on the data cable are divided into the following two categories:

1) Interface message

During communication between the instrument and the host computer, the attention cable shall be pulled down first, and then the interface message will be transmitted to the instrument through the data cable. Only instruments with GPIB bus function can send interface message.

2) Instrument message

For the detailed structure and syntax of the instrument message, refer to Section "2.1.3 SCPI". The instrument message can be divided into two types as per the transmission direction, namely, command and instrument response. Unless otherwise stated, all programmed control interfaces apply instrument message in the same way.

a) Commands:

Commands (programming messages) are messages sent by the host computer to the instrument for the remote control of instrument functions and the query of state information. Commands are divided into the following two categories:

- Based on the impact on the instrument:
  -- Setting commands: change the set state of the instrument, such as reset or setting frequency.
  -- Query commands: query and return data, for example: identify the instrument or query the parameter value. Query commands end with the suffix question mark.
- Based on the definition in the standard:
  -- common commands: with functions and syntax to be defined by IEEE488.2, they are applicable to all types of instruments (if realized)
  -- It is used for management of standard state register, resetting, self-detection, etc.
  -- Instrument control commands: instrument characteristic commands, used to realize instrument functions, such as setting frequency.
  The syntax also follows the specifications of SCPI.

b) Instrument responses:

Instrument responses (response message and service request) are the query result information sent by the instrument to the computer. Such

information includes measurement results, instrument state, etc.

2.1.3 SCPI

- Introduction to SCPI.....10
- Description of SCPI.....11

2.1.3.1 Introduction to SCPI

SCPI (standard commands for programmable instruments) are a command set for all instruments established based on Standard IEEE488.2. The main purpose is to make the same function have the same programmed control command to achieve the universality of programmed control commands.

SCPI consist of a command header and one or more parameters. The command header is separated from the parameters by spaces and contains one or more key fields. A command with direct suffix question mark is a query command. Commands are divided into common commands and instrument commands that have different syntactic structures. SCPI have the following characteristics:

1) Programmed control commands are oriented to test function rather than describing instrument operation;
2) Programmed control commands reduce the repetition of similar test function realization process, and ensure the compatibility of programming.
3) Programmed control messages are defined in layers that are hardware independent of the communication physical layer;
4) Programmed control commands are independent of programming methods and languages. The test program of SCPIs is easy to transplant;
5) Programmed control commands are scalable and can adapt to different scale of measurement control;
6) SCPI have been a “living” standard for their scalability.

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 programmed control command set, classification and description of the 87237, please refer to:
  1) "3. Programmed control commands" in this manual;
  2) “Annex A Zoom Table of SCPI commands classified as per subsystems” of this manual.

2.1.3.2 Description of SCPI

- General terms....11
- Command type....12
- Instrument command syntax....12
- Command tree....16
- Command parameters and responses....17
- Number system of commands....20
- Command line structure....21

1) General terms

The following terms apply to this section. To better understand the chapters, you shall understand the exact definitions of the terms.

Controller

A controller is any computer used to communicate with the SCPI device. A controller may be a PC, minicomputer, or a plug-in card on a cage. Some AI devices can also be used as controllers.

Device

A device is any device that supports SCPI. Most of the devices are electronic measurement or excitation devices that use GPIB interfaces for communication.

Programmed control message

A programmed control message is the combination of one or more SCPIs that have been correctly formatted. Programmed control messages tell the devices how to measure and output the signals.

Response message

A response message is a set of data of specified SCPI formats. Response messages always come from the devices to controllers or listening devices. Response messages tell the controllers about the internal state or measurement values of the devices.

Command

A command is an instruction that satisfies the SCPI standard. The combination of commands controlling the devices forms a message. In general, a command includes keywords, parameters, and punctuation.

Event command

Event-type programmed control commands cannot be queried. An event command generally has no corresponding front panel key setting, and its function is to trigger an event at a specific time.

Query

A query is a special type of command. When a control device is queried, a response message appropriate to the controller syntax requirements is returned. A query statement always ends with a question mark.

2) Command type

There are two types of SCPI: common commands and instrument commands. Figure 2.1 shows the difference between the two commands. Common commands, defined by IEEE 488.2, are used to manage macros and state registers and for synchronization and data storage. Because common commands all start with an asterisk, they are easy to be recognized. For example, *IDN? *OPC, *RST are all common commands. Common commands are not part of any instrument commands, and the instrument interprets them in the same way regardless of the current path setting of the commands.

Instrument commands are easy to be recognized because they contain a colon (:). A colon is used in the beginning of an expression or between two keywords, for example: FREQuency[:CW?]. According to the internal function module of the instrument, instrument commands are divided into sub-sets of corresponding subsystem commands. For example, the power subsystem (:POWer) contains power-related commands, while the state subsystem (:STATus) contains commands for the state control register.

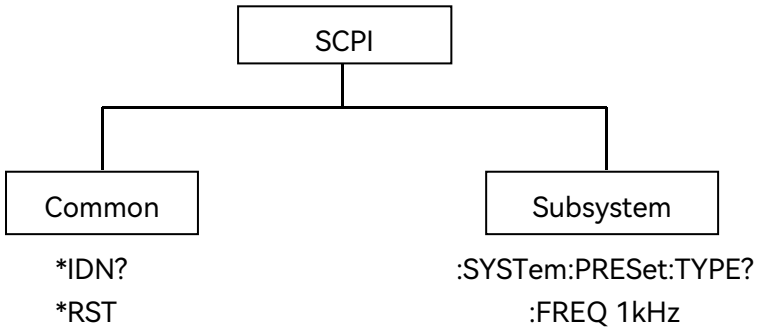


Figure 2.1 Types of SCPI

3) Instrument command syntax

A typical command consists of a keyword prefixed with a colon. The keyword is followed by parameters. The following is an example of a syntax declaration:

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

In the example above, the [:CW|FIXed] in the command follow : FREQuency closely without any space. MINimum|MAXimum immediately following [:LEVel] is the parameter. There is a space between the command and the parameter. Other parts of the syntax expression are described in Table 2.2 and Table 2.3.

Table 2.2 Special characters in command syntax

Sym	Meaning	Example
|	The vertical bar between the keyword and the parameter represents multiple options.	[:SENSe:]BANDwidth|BWIDth HIGH|LOWer BANDwidth and BWIDth are optional; HIGH and LOWer are optional.
[]	A square bracket indicates that the contained keyword or parameter is optional when forming a command. The command will be executed even when such implied keyword or parameter is ignored.	[:SENSe:]BANDwidth? SENSe is optional.
< >	The part in angle brackets indicates that the command is not used in the literal sense. They represent the part that must be contained.	[:SENSe:]FREQency[:CW|FIXed] <val>[unit] In this command, <val> must be replaced with actual frequency. [unit] is an omittable unit. For example: FREQ 3.5GHz FREQ 3.5e+009
{ }	The part in braces indicates that the parameter is optional.	MEMory:TABLe:FREQuency <val>{,<val>} For example: MEM:TABL:FREQ 5e7

Table 2.3 Command syntax

Characters, keywords and syntax	Example
Uppercase characters represent the minimum set of characters required to execute a command.	[:SENSe:]FREQuency[:CW|FIXed]?, FREQ is the short format part of the command.
Lowercase character of the command is optional; such flexible format is called "flexible listening". Refer to the section "Parameters and responses of commands" for more information.	:FREQuency :FREQ,:FREQuency or or :FREQUENCY Either of them is correct.
When a colon is between the two command mnemonics, it moves the current path in the	:TRIGger:MODE? TRIGger is the topmost keyword of

2.1 Remote control basics

Characters, keywords and syntax	Example
command tree down by one level. For more information, please refer to the command path part in section “Command tree”.	this command.
If the command contains more than one parameter, adjacent parameters are separated. The parameter is not part of the command path, so it does not affect the path layer.	MEMory:TABLe:FREQuency <val>{,<val>}
The semicolon is used to separate 2 adjacent commands, without affecting current command path.	:FREQ 2.5GHZ; :POW 10DBM
Blank characters, such as <space> or <tab>, are usually ignored as long as they do not appear between keywords or in keywords. However, you must separate the commands and parameters with blank characters, which does not affect the current path.	:FREQuency or :POWer :LEVel6.2 is not allowed. :LEVel and 6.2 must be separated by a space, namely, :POWer:LEVel 6.2.

The simplified syntax specification is shown in Figure 2.2.

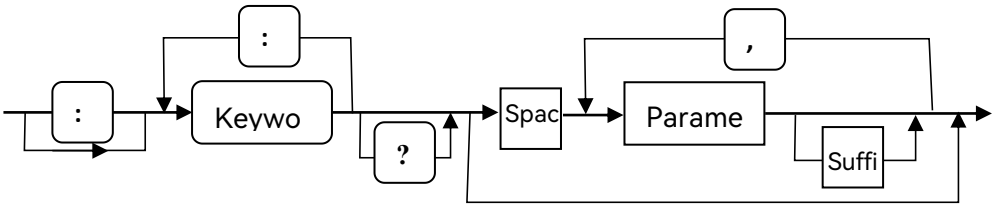


Figure 2.2 Types of SCPI

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

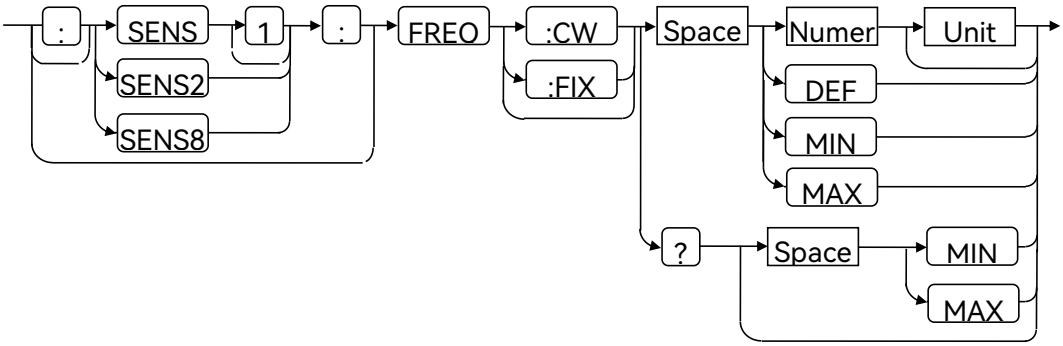


Figure 2.3 Types of SCPI

Remarks:

1) In the above chart, no spaces may be added where spaces are not indicated as required. Spaces may be 1 or more. If the units are omitted, the standard unit of frequency, Hz, and the standard unit of time, seconds, are used.

2) Rounded rectangles indicates the actual characters required for the keywords or commands, such as ":", ",", "?" , "1", etc.; Right-angled rectangles need to be replaced by actual characters, numbers, etc. For example, the "value" cannot appear in the command, and it needs to be replaced by the actual value, such as 5e+007, etc.

3) Without considering the short and long format of the keyword, the frequency of channel A is set to 50MHz, and the above chart has the following forms (only the short format of the keyword is taken. Since there are many units of frequency, such as Hz, kHz, MHz, GHz, THz, etc., it is no longer given one by one due to space limitations. It is only necessary to replace 5.0e+007 with corresponding units, such as 50MHz, 5e+007Hz, 0.05GHz, etc.).
   a) :SENS1:FREQ:CW 5.0e+007 No keywords omitted
   b) SENS1:FREQ:CW 5.0e+007 The ":" before SENS1 omitted.
   c) SENS:FREQ:CW 5.0e+007 "1" omitted.
   d) FREQ:CW 5.0e+007 SENS omitted.
   e) :SENS1:FREQ:FIX 5.0e+007 No keywords omitted
   f) SENS1:FREQ:FIX 5.0e+007 The ":" before SENS1 omitted.
   g) SENS:FREQ:FIX 5.0e+007 "1" omitted.
   h) FREQ:FIX 5.0e+007 SENS omitted.
   i) :SENS1:FREQ 5.0e+007 CW or FIX omitted
   j) SENS1:FREQ 5.0e+007 The ":" before SENS1 and CW or FIX omitted.
   k) SENS:FREQ 5.0e+007 "1" and CW or FIX omitted.
   l) FREQ 5.0e+007 SENS and CW or FIX omitted.

4) For channel B and channel 8 frequencies, SENS2 and SENS8 cannot be omitted.

5) MIN and MAX can be used as parameters of the set command or as parameters of the query command. DEF can only be used as a parameter of the set command. The specific values of MIN, MAX, and DEF are related to the instrument.
   a) FREQ DEF Set the frequency of channel A to the default value.
   b) FREQ? MAX Query the maximum settable frequency of channel A and return a value without unit.

6) If the long and short forms of keywords are considered, and commands with units are not considered, there are 1632 forms of commands in the above chart. Users don't have to care about all forms. As long as it can be used flexibly, it is enough. A simple calculation is performed below, and interested users can calculate by themselves.
   a) First calculate the case where SENS is defaulted, noted as N1.
      There are 2 forms of FREQ/FREQuency, noted as N11.
      There are four forms of CW/FIX/FIXed/Omitted, noted as N12.
      There are 7 parameters of the set command:
      Value/DEF/DEFault/MIN/MINimum/MAX/MAXimum.
      The parameters of the query command are

MIN/MINimum/MAX/MAXimum/Omitted, i.e., there are 12 types of parameters, which are recorded as N13, then

$$N1 = N11 \times N12 \times N13 = 2 \times 4 \times 12 = 96$$

b) Then calculate the case where SENS is not defaulted, noted as N2.
The first ":" has 2 types, omitted and not omitted, noted as N21.
There are 2 types of SENS/SENSe. There are 4 types of suffixes: 1/2/6/Omitted, i.e., there are 8 (2×4) types of SENS keywords, noted as N22.
SENS keyword forms a combination with the following keywords, then

$$N2 = N21 \times N22 = 2 \times 8 \times N1 = 16 \times N1$$

c) Let there be a total of N forms, then

$$N = N1 + N2 = 17 \times N1 = 1632$$

4) Command tree

Most remote control programs apply instrument commands. When parsing such commands, SCPI apply a file system-like structure called command tree, as shown in Figure 2.4:

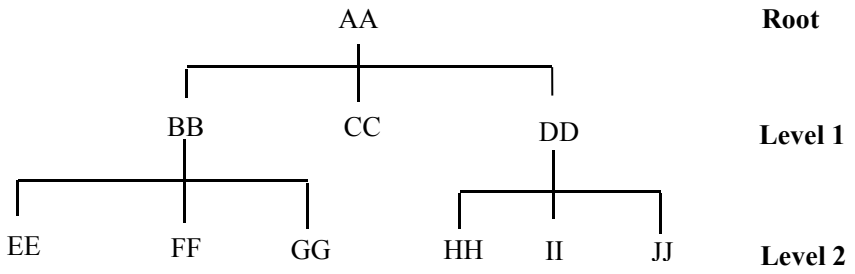


Figure 2.4 Diagram of the simplified command tree

The top command is the root command, or "root" for short. When a command is parsed, follow a specific path to the next level of command according to the tree structure. For example, in :POWer:ALC:SOURce?, : POWer stands for AA, :ALC stands for BB, :SOURce stands for GG, and the entire command path is (:AA:BB:GG).
A software module in instrument software - **command interpreter**, is responsible for parsing each received SCPI. The command interpreter breaks commands into individual command elements by using a series of rules that distinguish the path of the command tree. After parsing the current command, keep the current command path unchanged. The advantage of this is to parse subsequent commands more quickly and efficiently since that the same command keyword may appear in different paths. After booting or *RST (reseting) the instrument, current command path is reset to root.

5) Command parameters and responses

SCPI define different data formats in the use of programmed control and response messages to comply with the principles of "*flexible listening*" and "*precise speaking*". For more information, please refer to IEEE488.2. "*Flexible listening*" means that the formats of the commands and parameters are flexible.

For example, set the frequency offset state command :FREQuency:OFFSet:STATe ON|OFF|1|0.

The following command formats are all used to set the frequency offset function to ON:

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

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

Each parameter type has one or more corresponding response data types. During query, a data type will be returned for a numerical parameter, and the response data is precise and strict, known as "**precise speaking**."

For example, during query of the power state (:POWer:ALC:STATe?), when it is ON, the response data returned is always 1 during query, regardless of whether the previously sent setting command is :POWer:ALC:STATe 1 or :POWer:ALC:STATe ON.

Table 2.4 Parameter and response types of SCPI

Parameter type	Response data type
Numerical	Real number or integer
Extended numerical	Integer
Discrete	Discrete
Boolean	Digital boolean
String	String
Blocks	Finite-length blocks
	Infinite-length blocks
Non-decimal numeric types	Hexadecimal
	Octal
	Binary

Numerical parameters

Numeric parameters can be used in both instrument-specific commands and common commands. A numeric parameter receives all the usual decimal counting methods, including signs, decimals, and scientific notation. If a device only accepts a specified numeric type, such as an integer, it will automatically round up the received numeric parameters.

Examples of numeric parameters:

0 No decimal point

100 Optional decimal point
1.23 Signed bit
4.56e<space>3 Index marker e can be followed by a space
-7.89E-01 Index marker e can be uppercase or lowercase
+256 Positive lookahead allowed
.5 Decimal points can be used first

Extended numerical parameters

Most measurements related to instrument commands use extended numeric parameters to specify physical quantities. Extended numerical parameters receive all numeric parameters and additional special values. All the extended numeric parameters receive MAXimum and MINimum as parameter values. Whether other special values, such as UP and DOWN, will be received is determined by the ability of the instrument to parse. All effective parameters will be listed in the table of SCPI.
Attention: Extended numeric arguments do not apply to common commands or STATus subsystem commands.
Examples of extended numeric parameters:

101	Numeric parameter
1.2GHz	GHz can be used as an index (E009)
200MHz	MHz can be used as an index (E006)
-100mV	-100 millivolt
10DEG	10°
MAXimum	Maximum effective setting
MINimum	Minimum effective setting
UP	Increase by a step
DOWN	Reduce by a step

Discrete parameters

When the number of parameter values to be set are finite, they are identified by discrete parameters. Discrete parameters use mnemonics to represent each valid setting. Like programmed control command mnemonics, discrete parameter mnemonics have two formats, long and short, and allow for mixture of upper and lower cases.
In the following examples, discrete parameters and commands are used together.

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

BUS GPIB, LAN, RS-232 trigger
IMMediate Trigger immediately
EXTernal Trigger externally

Boolean parameters

A Boolean parameter represents a true or false binary condition, which can only have four possible values.

Examples of Boolean parameters

ON	Logically true
OFF	Logically false
1	Logically true
0	Logically false

String parameters

String parameters allow ASCII strings to be sent as parameters. Single quotes and double quotes are used as separators.
The following are example of string parameters:

'This is Valid' "This is also Valid" 'SO IS THIS'

Real response data

Most of the test data are of real number type, and their formats can be basic decimal notation or scientific notation, which are supported by most advanced programming languages.

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

An integer response data is a decimal expression of an integer value containing signed bit. When querying the state register, most of the response data returned are of integer type.
Examples of integer response data:

0	Symbol bit optional
+100	Positive lookahead allowed
-100	Negative lookahead allowed
256	No decimal point

Discrete response data

Discrete response data are basically the same as discrete parameters, only that the return format of discrete response data is only a short form in uppercase.
Examples of discrete response data:

INTernal	Stabilization mode is internal
EXTernal	Stabilization mode is external

MMHead	Stabilization type is millimeter wave source module

Digital bool response data

A binary value 1 or 0 is returned as Boolean response data.

String response data

String response data and string parameters are alike. The main difference is that the separators of string response data are double quotes instead of single quotes. Double quotes can also be embedded in string response data, and there may be no characters between the double quotes.
Here are some examples of string response data:

"This is a string"

"one double quote inside brackets: ("")"

Arbitrary data block

See Section 7.7.6 of IEEE 488.2 for <Arbitrary data block>.

#nNNN...Nddd……ddd<LF>

- Data newline character indicating the end of the data block.
- Data length (i.e., the number of bytes of d)
- Number of bits of data length (i.e., the number of bits of N)
- The marker for the start of the data block.

For example: #42004 ..... in <LF> n = 4 and N = 2004.

6) Number system of commands

The value of the command can be entered in binary, decimal, hexadecimal or octal format. When using binary, hexadecimal or octal format, a proper identifier is required before the value. The decimal format (the default format) does not require an identifier. When a value is entered without an identifier in front of it, the device will ensure it to be in decimal format. The following list shows the identifiers required for different formats:

- #B indicates that the number is a binary number;
- #H indicates that the number is a hexadecimal number;
- #Q indicates that the number is an octal number.

The following are various representations of the decimal number 45 in SCPI:

- #B101101
- #H2D
- #Q55

The following example sets the RF output power to 10dBm (or a value of the equivalent value of the currently selected unit, such as DBUV or DBUVEMF) with a hexadecimal value of 000A.

:POW #H000A

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

7) Command line structure

A command line may contain multiple SCPI. To indicate the end of the current command line, the following methods may be used:

- Line feed;
- Line feed and EOI;
- EOI and the last data byte.

Commands on the command line are separated by semicolons, and commands belonging to different subsystems begin with a colon. For example:

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

The command line contains two commands: the first one belongs to the MMEM subsystem, and the second belongs to the HCOP subsystem. If adjacent commands belong to the same subsystem with repeated command path, they can be expressed in abbreviation. For example:

HCOP:ITEM ALL;:HCOP:IMM

The command line contains two commands: both of them belong to the HCOP subsystem, with the same first level. Therefore, the second command can start from the next level of HCOP, and the colon for starting the command can be omitted. It can be abbreviated as follows:

HCOP:ITEM ALL;IMM

2.1.4 Command sequence and synchronization

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

- Sequential commands are sequences of commands that are executed continuously. Each command is usually executed faster;
- An overlapping command is one that is not executed automatically before the next command is executed. It usually takes longer to process overlapping commands, and programs are allowed to process other events synchronously during the period.

Even if there are multiple setting commands on a command line, they may not be executed in the order they were received. To ensure that commands are executed in a certain order, each command must be sent on a separate command line.

Example: the command line contains setting and query commands

If multiple commands on a command line contain query commands, the query results are unpredictable. A fixed value is returned for the following command:

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

Returned value: 1000000000 (1GHz)

The following command returns an unfixed value:

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

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

TIPS

Setting commands are sent separately from query commands

General rules: in order to ensure the correctness of returned results of query commands, setting commands and query commands shall be sent in different programmed control messages.

2.1.4.1 Prevention of overlapping execution of commands

In order to prevent overlapping execution of commands, multithreading or commands *OPC, *OPC? or *WAI may be applied, which are executed only after the hardware settings are completed. During programming, the computer may force a period of time to synchronize certain events. The descriptions are shown below:

- Controller program using multi-threading

  Multi-threading is used to wait for command completion and synchronization between the UI and programmed control, that is, to wait for *OPC? Completion in separate threading without interfering GUI or programmed control threading execution.

- The application of the three commands in synchronous execution is shown in the table below:

Table 2.5 Command usage

Method	Action	Programming method
*OPC	After the command is executed, set it in the operation completion bit of the ESR register.	Set to ESE BIT0; Set to SRE BIT5; Send overlapping commands and *OPC; Wait for service request (SRQ); Service request represents that the overlapping command has been executed.
*OPC?	Stop executing the current command 1. The command is returned only when it is in the operation completion bit of the ESR register, indicating	Terminate processing of the current command before executing other commands, and send the command directly after the current command.

	that the previous command has been processed.	
*WAI	Before the execution of *WAI, wait for all commands to be sent before proceeding with unfinished commands.	Terminate the processing of the current command before executing other commands, and send the command directly after the current command.

In the case that the processing time of overlapping command is short, the command *WAI or *OPC may be used after the overlapping command to achieve command synchronization. In order to execute other tasks synchronously while the computer or instrument is waiting for the completion of overlapping commands, the following synchronization techniques may be applied:

- OPC and service request

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

Service request represents that the overlapping command has been executed.

- OPC? and service request

1) Set the bit4 of SRE: *SRE 16 enable MAV service request;
2) Send overlapping commands and *OPC?;
3) Wait for service request.

Service request represents that the overlapping command has been executed.

- Event state register (ESE)

1) Set the OPC mask bit of ESE (bit0): *ESE 1;
2) Send only overlapping commands instead of *OPC, *OPC or *WAI;
3) Send "*OPC;*ESR?" in timer for cyclic query of the operation state.

The returned value (LSB) 1 indicates completion of the overlapping command.

- * OPC? and short timeout

1) Send only overlapping commands instead of *OPC, *OPC or *WAI;
2) Send "<short timeout>; *OPC?" in timer for cyclic query of the operation state;
3) The returned value (LSB) equal to 1 indicates that the execution of the interleave command is complete. In case of timeout, it is during operation.
4) Reset the timeout value to the old value;
5) Send the command "SYStem:ERRor?" to clear the error queue, and delete the message "-410, query interrupt".

The returned value (LSB) 1 indicates completion of the overlapping command.

2.1.5 State reporting system

The state reporting system stores all operation state information for the current

instrument, including error message. They are stored in state registers and error queues respectively, and can be queried through a programmed control interface.

- Structure of state register 24
- Structure of SCPI state register 25
- Description of state register 26
- Application of state reporting system 29
- Resetting state reporting system 31

2.1.5.1 Structure of state register

State registers are described by classification below:

1) STB, SRE

The state byte (STB) register and its related mask register – service request enable register (SRE), comprise the top register of the state reporting system. STB saves the general operation state of the instrument by collecting the information of lower registers.

2) ESR, SCPI state register

STB receives information from the following registers:

- The value of the event state register (ESR) and the event state enable (ESE) mask register.
- SCPI state register includes: STATus:OPERation and STATus:QUEStionable registers (SCPI definition), which contain the specific operation information of the instrument. All SCPI state registers have the same internal structure (For details, refer to 2.1.5.2 “SCPI state register structure” of the programmed control manual).

3) IST,PPE

Similar with SRQ, IST (“Individual STatus”) marks a separate bit consisting of all states of the instrument. The associated parallel poll enable register (PPE) determines the STB data bits for IST marking.

4) Output buffer

It stores the messages returned by the instrument to the controller. It does not belong to the state report system, but determines the MAV position value of the STB.

TIPS

SRE, ESE

SRE may be used as the enable part of STB. Similarly, ESE may be used as the enable part of ESR.

2.1.5.2 SCPI state register structure

Each standard SCPI register consists of five parts. Each part contains 16 bits and is functionally independent. For example, one bit is assigned for each hardware state and valid for all five parts of the register. If Bit15 is set to 0, the value of the register is positive integer data.

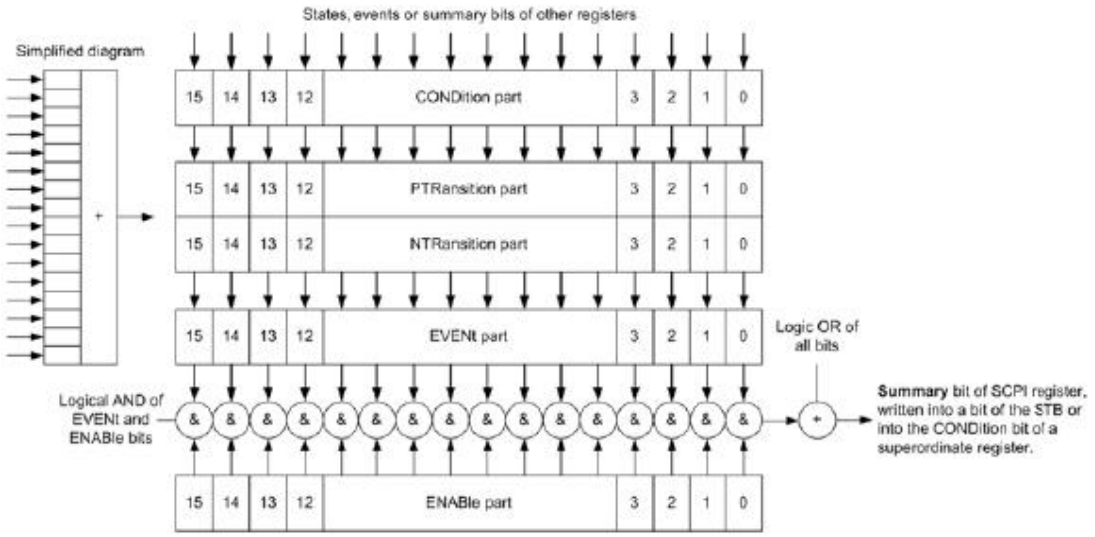


Figure 2.5 Structure of state register

It can be seen from the figure above that the state register consists of five parts, which are respectively described as follows:

- Condition register

In this part, the summary bit of hardware or lower registers are directly written, reflecting the operation state of the current instrument. This register is read only, not writable. It reads but not clearing the value.

- Positive/negative transition register

The two transition registers define the state transition bit of the condition register stored in the event register.

The positive transition register is similar to the transition filter. When a bit of the condition register is transformed from 0 to 1, relevant PTR bit determines whether the event bit is set to 1, as shown below:

-- PTR bit = 1: the event bit is set.

-- PTR bit = 0: the event bit is not set.

The positive transition register is readable and writable. It reads but not clearing the value.

The negative transition register is similar to the transition filter. When a bit of the condition register is transformed from 1 to 0, relevant NTR bit determines whether the event bit is set to 1, as shown below:

-- NTR bit = 1: the event bit is set.

-- NTR bit = 0: the event bit is not set.

The positive transition register is readable and writable. It reads but not clearing the value.

- Event register

This part indicates whether the event has occurred since the last reading and whether the content of the condition register is stored. It represents only the event passed through the transition register. It 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 often equal to the value of the entire register.

- Enable register

This part determines whether the related event bit acts on the final summary data. The bits of each enable part is the sum of related enable bits. The logical operation result of this part is or not the summary bit.

- enable bit = 0: the related event bit does not act on the summary data.
- enable bit = 1: the related event bit acts on the summary data.

This part is readable and writable. It reads but not clearing the value.

- Summary bit

The summary bit of each register consists of the event and the enable part. The result enters the condition part of the upper register. The instrument automatically generates the summary bit for each register so that events can cause different levels of service requests.

2.1.5.3 State register description

The state registers are detailed as follows:

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

IEEE488.2 defines the state byte (STB). The rough instrument state is reflected by collecting the information of lower registers. Bit6 is equal to the summary data of other state byte bits. The result of a comparison between the state byte and the condition part of the SCPI register may be assumed to be the top in the SCPI hierarchy. The value of state byte may be read through common command "*STB?" or serial query.

The state byte is connected to the service request enable register (SRE). Each bit of the state byte corresponds to a bit in SRE. Bit6 of SRE is ignored. If one of the bits in SRE is set and the related STB bit changes from 0 to 1, a service request (SRQ) will be generated. Common command "*SRE" is used to set SRE, and common command "*SRE?" used to read SRE. The state byte is described in Table 2.6 Description of state byte:

Table 2.6 Description of state byte

Bit	Meaning
0	Not used.
1	Device related.

Bit	Meaning
2	The error queue is not empty Set this bit when a new error is inserted into the error queue. If related SRE bit enables the bit, a service request will be generated when a new error is generated in the error queue, so that the error can be identified and the error message can be queried. Such method effectively reduces errors in programmed control.
3	Summary bit of question state register Set this bit only when the event bit of the inquiry state register and the related enable bit are set to 1. This bit represents a question state of the instrument. Specific instrument state information can be obtained by querying the question state register of the state register.
4	MAV bit (message available) Set this bit if the output queue information is readable. This bit is used when the controller queries instrument information.
5	ESB bit Summary bit of the event state register. Set this bit when one of the bits in the event state register is set and the corresponding bit in the event state enable register is enabled. The bit of 1 indicates a serious error in the instrument. The specific error message can be found by querying the event state register.
6	MSS bit (master status summary bit) Set this bit if the instrument triggers a service request.
7	Summary bit of operation state register Set this bit when the event bit of the operation state register and the corresponding enable bit are set to 1. This bit indicates that the instrument has performed an operation, the type of which can be obtained by querying the operation state register.

2) IST flag and parallel poll enable (PPE) register

The IST identifies the combination of the overall instrument state with a single data bit. The flag can be obtained by a parallel poll or by sending the command "*IST?". The associated parallel poll enable register (PPE) determines the STB data bits for IST marking. STB data bits are in phase with PPE data bits, and the usage of bit6 is opposite to bit6 in SRE. The IST flag is equal to the limit of all results. The command "*PRE" may be used to set PPE, and the command "**PRE?" used to read PPE.

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

IEEE488.2 defines ESR. The command "*ESR?" may be used to read the event state register (ESR). ESE belongs to the enable part of SCPI register. If one of the bits is 1 and one of the bits in the corresponding ESR changes from 0 to 1, the ESB bit of STB shall be

set to 1. The command "*ESE" may be used to set ESE, and the command "*ESE?" used to read ESE.

Table 2.7 Description of event state byte

Bit	Meaning
0	Operation completed Set this bit when the previous commands have been executed and the command *OPC has been received.
1	Not used.
2	Query error Set this bit when the controller reads the instrument data without sending the query command, or sends a new command before reading the query data. It indicates that there is a query error, for which the query cannot be executed.
3	Instrument error Set this bit when there is an instrument error. Error code range: -300 - -399, or positive error code. Specific error message can be found in relevant information in the error queue.
4	Execution error Set this bit when a syntactically correct command is received but cannot be executed, and an error with code ranging from -200 to -300 is generated in the error queue.
5	Command error Set this bit when the syntax of the command received is incorrect. Error code range: -100 - -200. Specific error message can be found in relevant information in the error queue.
6	User request Set this bit when the instrument is switched to manual control mode.
7	Power ON Set this bit when the instrument is powered on.

4) State: operation register

State: The operation register contains information about the current operation of the instrument and information about the previously executed operations. The operation register value can be read by the command "STATus:OPERation:CONDition?" or "STATus:OPERation[:EVENt]?" to read the operation register value. The register is described in Table 2.8 below.

Table 2.8 State: operation register description

Bit	Value	Definition
0	1	Unused

1	2	Channel A calibration state
2	4	Channel B calibration state (for dual-channel only)
3	8	USB interface channel calibration state
4-14	-	Unused
15	-	Always 0

5) State: question register

The register contains instrument state that does not meet specification requirements. The register value may be queried through the command "STAT:QUES:COND" or "STAT:QUES:EVEN". The register is described in Table 2.9 below.

提 示

Query register

State: the question register has collected the information of all lower sub-registers (for example, bit2 has collected all time related information). Since each path corresponds to a separate sub-register, in case of a state bit error of the question register, it is required to go back to the sub-register of the path to check for the specific error source. By default, the sub-register state being queried belongs to the currently selected path.

Table 2.9 State: question register description

Bit	Value	Definition
0-2	-	Unused
3	8	Power Summary
4-7	-	Unused
8	256	Calibration Summary
9	512	Power-on self-test
10-14	-	Unused
15	-	Always 0

2.1.5.4 Application of state reporting system

The state reporting system is used to monitor the state of one or more instruments in a test system. In order to correctly realize the function of the state reporting system, the controller in the test system must receive and evaluate the information of all instruments. Standard methods used include:

1) Service request (SRQ) initiated by the instrument;
2) Serial query of all instruments in the bus system, initiated by the controller in the system, in order to find the initiator of the service request and the reason.
3) Parallel query of all instruments;

4) Programmed control command to query the state of specific instruments;
5) Query of error queue.

1) Service request

In some cases, the instrument sends a service request (SRQ) to the controller to obtain the controller's service, and the controller initiates an interrupt to enter the corresponding interrupt handler. According to Figure 2.5, an SRQ is typically initiated by one or more state bytes and by bits 2, 3, 4, 5 or 7 of the related enable register (SRE). These bits, in turn, make up advanced registers, error queues or output buffers. In order to use all the service requests as far as possible, all bits in enable registers SRE and ESE shall be set to 1.

Example: use the command *OPC to generate SRQ at the end of the sweep.

a) Call the function InstrWrite to write the command "*ESE 1", and set to ESE bit0 (operation completed).
b) Call the function InstrWrite to write the command "*SRE 32", and set to SRE bit5 (ESB).
c) Call the function InstrWrite to write the command "*INIT;*OPC", and SRQ is generated after the operation is completed.

After instrument setting, the instrument generates a SRQ.

SRQ can only be initiated by the instrument. In case of an instrument error, the controller program shall allow a service request to be made to the instrument and handled by a dedicated interrupt service program.

2) Serial query

Similar to the command *STB, serial query is used to query the state byte of the instrument. Serial query adopts the method of interface message, so the query speed is fast. IEEE 488.2 defines the specific method for serial query. The method is mainly used to quickly obtain the state of one or more instruments connected with the controller in the test system.

3) Parallel query

In the test system, the controller sends an information bit to the data cable through a command, and can query 8 instruments at the same time. The data configured on the data cable of the instrument is a logical "0" or "1". In addition to the conditions under which the SRE register determines the SRQ to be generated, the bits of parallel poll enable register (PPE) and STB register shall be subject to AND operation. The result obtained is sent to the controller of parallel query as the response result after OR operation and NOT operation, or the result may be obtained through the command *IST.

In parallel query, first the instrument shall be set to the parallel query state through the command PPC, which allocates one data cable to the instrument and determines whether the bit is reversed in response. The PPE register is used when executing parallel

query. Parallel query is mainly used for the controller to quickly locate which instrument has sent the service request. Therefore, the same values shall be set for the registers SRE and PPE.

4) Query instrument state

The following two commands may be used to query each part of the state register:

- Command *ESR?, *IDN?, *IST?, *STB? is used to query the advanced register;
- The state system command is used to query the SCPI register (for example: State:QUEStionable…).

The returned value of the register being queried is usually in decimal format and is detected by the controller program. For more details on why SRQ is generated, parallel query is usually done after SRQ.

Description of response data bit

The STB and ESR registers contain 8 bits, and the SCPI register contains 16 bits. The returned value of the query state register is in decimal format. The decimal value is equal to the sum of each bit and respective weight.

The relationship between the bit and the weight is shown in the figure below:

数据位	7	6	5	4	3	2	1	0
权 重	128	64	32	16	8	4	2	1

Figure 2.6 Relationship between the bit and the weight

5) Error queue

Each error state of the instrument corresponds to an entry in the error queue, which contains a specific error message text that can be viewed through the error log or queried through the programmed control command: SYSTem:ERRor[:NEXT]? or SYSTem:ERRor:ALL?. If there is no error in the error queue, the query returns 0, "No error".

The error queue shall be queried in the controller service request handler because a more accurate description of the cause of the error can be obtained than in the state register. Especially in the test phase of the controller program, the error queue shall be frequently queried to clarify the error command record sent by the controller to the instrument.

2.1.5.5 Resetting the state reporting system

Commands and events for the resetting state reporting system are listed below. In addition to the commands *RST and SYSTem:PRESet, other commands will not change the function settings of the instrument. Similarly, DCL will not change the set state of the

instrument. Details are shown in the table below:

Table 2.10 Resetting the state reporting system

Event / Funct	Power ON/OFF (Powered state cleared)		DCL, SDC (Instrument cleared, instrument selected to be cleared)	*RST or SYSTem: PRESet	STATus: PRESet	*CLS
	0	1				
Clear STB, ESR	—	Yes	—	—	—	Yes
Clear SRE, ESE	—	Yes	—	—	—	—
Clear PPE	—	Yes	—	—	—	—
Clear the event part of the register	—	Yes	—	—	—	Yes
Clear the enable part of the operation and question registers. Fill the enable part of other registers with 1.	—	Yes	—	—	Yes	—
Fill the positive transition part with 1. Clear the negative transition part.	—	Yes	—	—	Yes	—
Clear the error queue	Yes	Yes	—	—	—	Yes
Clear the output buffer	Yes	Yes	Yes	—	—	—
Clear the command processing and input buffers	Yes	Yes	Yes	—	—	—

2.1.6 Precautions for programming

1) Please initialize the instrument state before changing settings

When setting the instrument remotely, first initialize the instrument state (for example, send "*RST"), and then implement the required state setting.

2) Command sequence

In general, setting commands and query commands shall be sent separately. Otherwise, the returned value of query commands will change with the current order of instrument operation.

3) Fault response

Service requests can only be initiated by the network analyzer. The controller program in the test system shall guide the instrument to initiate service request actively when there is an error, and then enter corresponding interrupt service program for processing.

4) Error queue

Each time the controller program processes a service request, it shall query the error queue of the network analyzer instead of the state register for a more precise cause of the error. Especially in the test phase of the controller program, the error queue shall be frequently queried to obtain the error command sent by the controller to the network analyzer.

2.2 Instrument programmed control port and configuration

The USB programmed control system uses USBTMC to control the 87237.

The USBTMC interface does not require user configuration.

NOTICE

Use of USB programmable interface

Before programming with this instrument, it is necessary to install VISA library in the host computer.

2.3 I/O library

- Overview of I/O Library.....33
- Installation and Configuration of I/O Library.....35

2.3.1 Overview of I/O library

I/O library is a pre-written software library for instruments, known as instrument driver. As a software between the computer and the instrument hardware, it consists of the function library, utility program, tool kit, etc. It is a combination of a series of software code modules and corresponds to operation of a plan, such as configuring the instrument, reading from the instrument, writing to the instrument and trigger the instrument, etc. Residing in the computer, it is the bridge and link between the computer and the instrument. By providing a high-level modular library for convenient

programming, users no longer need to learn the complex low-level programming protocol for a specific instrument. Application of instrument driver is the key to develop test and measurement applications quickly.

Functionally, a universal instrument driver generally consists of five parts: functor, interactive developer interface, programmer interface, subprogram interface and I/O interface, as shown in the figure below:

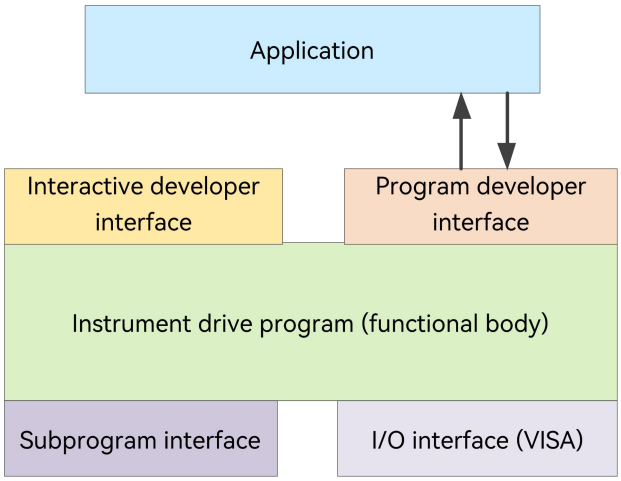


Figure 2.7 Structure model of instrument driver

The details are as follows:

1) Functor. As the main function part of the instrument driver, it may be understood as its framework program.
2) Interactive developer interface. Application development environment that supports instrument driver development is usually provided with graphical interactive developer interface for user convenience. For example, in Labwindows/CVI, the function panel is an interactive developer interface. In the function panel, each parameter of the instrument driver function is represented as a graphical control.
3) Programmer interface. It is a software interface for the application to call instrument driver function, such as dynamic link library file.dll of instrument driver in Windows system.
4) I/O interface. It completes the actual communication between the instrument driver and the instrument. The bus-specific I/O software, such as GPIB and RS-232, or the common standard I/O software applied cross multiple buses (VISA I/O) can be used.
5) Subprogram interface. It is a software interface for the instrument driver to access other support libraries, such as databases, FFT functions, etc. The subprogram interface is used when the instrument driver needs to call other software modules, operation systems, programmed control code libraries and analysis function libraries to complete its task.

2.3.2 Installation and configuration of I/O library

Along with the application in test field, it has gone through different development stages from traditional instrument to virtual instrument. In order to solve the interchangeability of instruments and reusability of test program in automatic test system, instrument driver has gone through different development processes. IVI (Interchangeable Virtual Instruments) driver is relative popular and common at present. Based on IVI specification, it defines a new instrument programming interface, inserts the class driver and VPP architecture onto the VISA to make the test application and instrument hardware completely independent, adds such unique functions as instrument simulation, range sensing and state cache, improves the operation efficiency of the system, and realizes instrument exchange.

There are two types of IVI driver: IVI-C and IVI-COM, where the latter adopts the form of COM API based on the component object model (COM) of Microsoft, and the former adopts the form of C API based on ANSI C. Both types are designed according to the instrument class defined in the IVI specification and have the same application development environment, including Visual Studio, Visual Basic, Keysight VEE, LabVIEW, CVI/LabWindows, etc.

TIPS

Port configuration and IO library installation

Before using the computer to control the 87237, please make sure you have the necessary ports and I/O libraries installed and configured correctly.

TIPS

Use of I/O library

The driver function panel, help document and driver function examples will be installed automatically when installing the attached IVI-COM/C driver installation package, so as to facilitate users to develop integrated programmed control functions.

2.4. Zeroing

Zeroing is used to deduct channel noise. When shall zeroing be carried out? It is recommended that the 87237 be zeroed in the following cases:

- When the temperature change exceeds 5℃;
- After the connection is established with the device under test;
- Shutdown and reboot;
- After 24 hours;

- Before measuring the low-power signal. For example, when measuring a signal 20dB higher than the minimum power specified in the 87237.

Related programmed control commands:

CALibration:ZERO:AUTO

2.5 Performing measurements

The measurement can be configured as absolute power measurement, differential power measurement, ratio power measurement, relative power measurement, etc. For details, refer to "3.3.3 Measurement sub-system commands".

2.6 Using frequency response offset table

- Overview.....36
- Entering frequency offset table..... 37
- Selecting frequency offset table..... 37
- Enabling frequency offset table..... 38
- Measurement applications..... 38
- Specific applications.....38

How to use the frequency response offset (hereinafter referred to as frequency offset) table? The frequency offset table is used to compensate the frequency response in the process of measurement establishment.

2.6.1 Overview

Enable or disable the frequency offset table with [SENSe[1]:]CORRection:CSET2:STATe. When enabled, the frequency offset table provides a quick way to compensate for the frequency response in the test system. It shall be noted that when enabled, the frequency offset is an "additional" frequency response to 87237, that is, the inherent frequency response of the 87237 (stored in the FLASH of the 87237) is also taken into account. The 87237 can store up to 10 frequency offset tables, each with up to 80 frequency points.

How to use the frequency offset table:

a) Enter frequency offset table

b) Select frequency offset table

c) Enable frequency offset table

d) Measurement applications

2.6.2 Entering frequency offset table

a) Entry steps
   - Enter the frequency list: MEMory:TABle:FREQuency <frequency 1>{, <frequency i>}. such as 50 MHz, 1 GHz, 10 GHz, 40 GHz;
   - Enter the offset factor corresponding to the frequency list: MEMory:TABle:GAIN <factor 1>{, <factor i>}. For example, 100, 98.8, 101.2, and 110.8 correspond to the offset factors of 50 MHz, 1 GHz, 10 GHz and 40 GHz, respectively;
   - Rename the frequency offset table if necessary: MEMory:TABLe:MOVE <original name>, <target name>. For example "User_3", "MyFdo0"

b) Enumerate the name of the frequency offset table: MEMory:CATalog:TABLe? see the description of this command for more information.

c) Rename the frequency offset table: MEMory:TABLe:MOVE <original name>, <target name>.

d) Query the data in the frequency offset table. For example, query the data in the frequency offset table with sequence number 3.
   - Query the frequency points in the frequency offset table:MEMory:TABLe3:FREQuency:POINTs?
   - Query the frequency list in the frequency offset table:MEMory:TABLe3:FREQuency?
   - Query the number of factor points in the frequency offset table: MEMory:TABLe3:GAIN[:MAGNitude]:POINTs?
   - Query the factor list in the frequency offset table: MEMory:TABLe3:GAIN[:MAGNitude]?

e) Modify the data in the frequency offset table: see a).

f) Notes
   - The frequency list is arranged in ascending order;
   - The effective suffix of the frequency list is Hz, kHz, MHz and GHz. Hz by default;
   - Ensure that the frequency list is within the effective frequency range of the 87237;
   - The name of the frequency offset table is up to 12 bytes and no spaces are allowed.

2.6.3 Selecting frequency offset table

[SENSe[1]:]CORRection:CSET2[:SELect] <frequency offset table name>, e.g. "User_3"

2.6.4 Enabling frequency offset table

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

2.6. 5 Measurement applications

- ABOR
- CONF:POW:AC DEF,1,(@1)
- SENS:CORR:CSET2:SEL "MyFdo0"
- SENS:CORR:CSET2:STAT ON
- SENS:FREQ 5GHz
- INIT1
- FETC?

2.6.6 Specific Applications

a) Suppose the frequency list of frequency offset table is: 500MHz, 1GHz, 11GHz. The list of offset factors is: 100, 10, 10. Let the signal frequency be Freq and the calculated offset factor be Gain.

b) If Freq is outside the range of the frequency offset table, the frequency bias value of the highest or lowest frequency point in the frequency offset table is used. For example, for 18 GHz, the offset factor 102 corresponding to the maximum frequency point of 10 GHz is used. For example, for 50MHz, the offset factor 100 corresponding to the minimum frequency point of 500MHz is used.

c) If Freq is within the valid range of the frequency list, but between two frequencies (Freq1, Freq2), such as 5 GHz, the offset factor is obtained using two-point linear interpolation, and the offset factors corresponding to Freq1 and Freq2 are set to Gain1 and Gain2, respectively. Gain is calculated to be 50 by the following equation.

$$Freq\,2 - Freq\,1$$

d) If the power before using the frequency offset table is 1.000mW (noted as Pwr0), and the final displayed power is Pwr, then Pwr = Pwr0/Gain/100 = 2.000mW.

2.7 Setting Display Resolution

Related programmed control commands:
DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution <resolution>
< Resolution> ranges from 1 to 4.
For linear power display, <resolution> represents the number of valid digits

displayed; for logarithmic power display, <resolution> represents the number of digits after the decimal point.

2.8 Setting average

The 87237 memory has a digital filter for average power readings. The average times ranges from 1 to 262144. If the average state is turned on, the measurement time will be increased.

Set to auto-average state, that is, set different averaging times according to different power level and display resolution. Generally, the lower the power, the greater the average times; the higher the resolution, the larger the average times.

Related programmed control commands:

[SENSe[1]:]AVERage[:STATe] <ON/OFF> Set the ON/OFF state of average

For example, turn on the measurement average:

SENS:AVER 1

Turn off the measurement average.

AVER 0

[SENSe[1]:]AVERage:COUNt:AUTO <ON/OFF> Set the ON/OFF state of auto-average

[SENSe[1]:]AVERage:COUNt <average times> Set average times

2.9 Setting Offset

The 87237 can compensate for signal attenuation or gain in test equipment (e.g., compensate for a 20 dB attenuator).

Related programmed control commands:

[SENSe[1]:]CORRection:GAIN2 <offset value> Set channel offset value

[SENSe[1]:]CORRection:GAIN2:STATe <ON/OFF> Set the ON/OFF state of channel offset

For example, set the channel offset to 3dB.

SENS:CORR:GAIN2 3

2.10 Setting measurement limits

Verify that the measured power is not outside the given range by setting the measurement limits.

Related programmed control commands:

:CALCulate[1]|2|3|4:LIMit:LOWer[:DATA] <lower limit> Set lower limit

:CALCulate[1]|2|3|4:LIMit:UPPer[:DATA] <upper limit> Set upper limit

:CALCulate[1]|2|3|4:LIMit:STATe <ON/OFF> Set the ON/OFF state of limit detection

:CALCulate[1]|2|3|4:LIMit:FAIL?	Query whether the limit is exceeded
:CALCulate[1]|2|3|4:LIMit:FCOunt?	Number of times to query whether the limit is exceeded (FCO)
:CALCulate[1]|2|3|4:LIMit:CLEar[:IMMediate]	Clear the failure count (FCO).

2.11 State report

The state report is used to detect the error information, operation state, question state, etc. of the power. The state report adopts the principle of step-by-step reporting, such as calibration operation is reported to operation state, and operation state is reported to state word.

For example, query whether the instrument is being calibrated, whether the instrument is being finished calibrating, whether there is an error in zeroing and calibration, whether the instrument is connected to 87237, whether zeroing is required, and whether the measurement is out of range, etc. The following is a description of "querying whether the instrument is calibrating channel A".

Bit 1 in the register group of STATus:OPERation:CALibrating indicates the calibration state of channel A.

Set the event occurrence of the calibration operation to detect the event when there is a transition from uncalibrated (state 0) to calibrated (state 1), i.e., set the positive transition filter to 1.

STAT:OPER:CAL:PTR 2(Configure the calibration event detection for channel A to be from 0 to 1) Note 1

Report calibration operation events to its higher-level operation state register. That is, the calibration state bit (bit 0) of the STATus:OPERation register group.

STAT:OPER:CAL:ENAB 2 (The parameter can be 4, or 6 if it is necessary to report events for channel B). Note 1

Similarly, if it is necessary to further report the operation state to bit 7 of the "state word", set the corresponding bit (bit 0) of the operation positive transition filter and the corresponding bit (bit 0) of the enable register.

STAT:OPER:CAL:PTR 1(Configure the calibration event detection to be from 0 to 1)

STAT:OPER:ENAB 1 (The parameter can be 2048, or 2049 if the lower detection event needs to be reported).

At this point, the configuration is complete, and the next step is to query whether channel A is being calibrated.

Method 1: Query bit 1 of the calibration operation condition register. If it returns non-zero, it is being calibrated.

STAT:OPER:CAL:COND?

Method 2: Query bit 0 of the operation condition register. If it returns non-zero, it is being calibrated.

STAT:OPER:COND?

Method 3: Query bit 7 of the state word. If it returns non-zero, it is being calibrated.

*STB?

Note 1: If it is necessary to query whether one of the two channels is being calibrated, change parameter 2 to 6 (bit 1 and bit 2 indicate channel A and channel B, respectively).

2.12 Storage call

To reduce repetitive setup processes, the instrument can store 10 types of configuration data into non-volatile memory. Error list, program-controlled address (such as IP address, instrument string number), frequency response offset table, zeroing and calibration information, etc. are not stored in this configuration. Except for the error list, which is not stored, everything else is stored in the hard configuration file and does not change with user calls.

Related programmed control commands:

*SAV <NRf>

*RCL <NRf>

<NRf> ranges from 1 to 10.

3. Programmed control commands

- Description of commands.....43
- Common commands.....43
- Instrument subsystem commands.....50

3.1 Description of commands

This section provides detailed command reference information for remote control, including:

- Complete syntax format and parameter list;
- Syntax diagram for non-standard SCPI;
- Detailed function description and related command description;
- Supported command formats (settings or queries);
- Parameter description, including: data type, value range and default value (unit);
- Key path;
- Model of instrument in the same class of instrument that is compatible with the command. If not specified, the current command only applies to this series.
- Other instructions.

The sections of common commands and instrument subsystem commands first list the order of command items to make convenient for users to query.

For remote control, the following table describes the command suffixes:

Table 3.1 Description of command suffixes

Suffixes	Value range	Description
<ch>	1..1	Channel
<m>	1..4	Measurement
<t>	1..10	Frequency response offset table

3.2 General commands(IEEE488.2 Commands)

Common commands are used to control instrument state registers, state reports, synchronization, data storage and other common functions. The use and function of common commands apply to different instruments. All common commands may be identified by the first "*" in the command word, and are defined in detail in IEEE488.2. IEEE488.2 common command is interpreted and explained below.

- *CLS.....44
- *DDT.....44
- *ESE.....44
- *ESR?.....46
- *IDN?.....46

3.2 General commands(IEEE488.2 Commands)

- *OPC 46
- *OPT 47
- *RCL 47
- *RST 47
- *SAV 49
- *SRE 48
- *STB? 48
- *TRG 49
- *TST? 49
- *WAI 49

TIPS

Command use:

Unless otherwise specified, commands may be used for settings or queries.

If a command is used only for settings or queries, or to start an event, the command description will be explained separately.

*CLS

Function: Clear the instrument state data structure, including SCPI registers (such as questior state, operation state, etc.), standard event registers, state words, and error/event queues.

Query: Not supported

Setting: *CLS

Example: *CLS Clear the instrument state

Error message: No

Reset state: No

*DDT

Function: Query or set the operation in response to *TRG general command.
Note: The 87237 does not support this command at the moment. This command is used for expansion.

Query: *DDT?

Setting: *DDT <Arbitrary data block> | <string>
Arbitrary data block is of the form: #nN<action>
The string will be of the form "<action>"
1) Action has the following forms:
FETC?
FETC1?

FETC2?
*TRG
TRIG1
TRIG2 (for dual-channel power meter only)

2) In arbitrary data block, the first value n after # represents the number of bits of the data length, and the next value represents the length of the data block.
Example: #15FETC? -- n = 1, N = 5 (5 bytes in total for FETC?)

Example:

*DDT?	Query the operation behavior of the instrument when it receives the *TRG command.
*DDT #206FETCh?	Set instrument to execute "FETCh?" upon receiving *TRG command Command
*DDT "FETCh?"	Set instrument to execute "FETCh?" upon receiving *TRG command Command
*DDT "TRIG1;FETC1"	Set the instrument to execute the "TRIG1;FETC1" command upon receiving the *TRG command.

Error message: No

Reset state: No

*ESE

Function: Query or set the standard event state enable register. 0 for disable, and 1 for enable.

Query: *ESE?

Setting: *ESE <NRf>

NRf denotes the numerical value, an integer between 0 and 255. See Table 3.2 for bit mapping.

Example:

*ESE?	Query the current setting of this register. The return format is <NR1>, 0-255.
*ESE 60	Enable 4+8+16+32 corresponding bits.

Error message: No

Reset state: No

Table 3.2 Standard event bit mapping

Bit	Value	Description
0	1	Operation completed
1	2	Unused
2	4	Query error
3	8	Device-related errors

4	16	Execution error
5	32	Command error
6	64	Unused
7	128	Unused

*ESR?

Function: Query the value of the standard event state register and clear the register. Refer tc Table 3.2

Query: *ESR?

Setting: Not supported

Example: *ESR? Query the value of the standard event state register and clear it.

Error message: No

Reset state: No

*IDN?

Function: Query the identification string of 87237.
Related programmed control commands ":SYSTem:IDN", ":SYSTem:IDN:AUTO"
If ":SYSTem:IDN:AUTO" is set to "ON", *IDN? returns the user string defined by ":SYSTem:IDN", otherwise returns the string preset by the 87237.

Query: *IDN?

Setting: Not supported

Example: *IDN? Query the identification string of 87237.

Error message: No

Reset state: No

*OPC

Function: When all waiting operations are completed, set the operation end bit in the standard event state register.

Query: *OPC?

Setting: *OPC

Example: *OPC? Return 1 if the waiting operation completes, otherwise it waits.

Error message: No

Reset No

state:

*ESE?

Function: Query the instrument option configuration.
Query: *ESE?
Setting: Not supported
Example: *ESE? Query the instrument option configuration.
Error message: No
Reset state: No

*RCL

Function: Call the 87237 state in the specified storage call register.
Query: Not supported
Setting: *RCL <NRf>
It is an integer ranging from 1 to 10
Example: *RCL 8 Call the instrument state from register 8.
Error message: If the registers are not located from 1 to 10, it prompts "-222, "Data out of range"".
Reset state: No

*RST

Function: To reset 87237, please refer to SYSTem:PRESet.
Query: Not supported
Setting: *RST
Example: *RST Reset
Error message: No
Reset state: No

*SAV

Function: Store the instrument state into the specified register.
Query: Not supported
Setting: *SAV <NRf>
It is an integer ranging from 1 to 10
Example: *SAV 9 Store the instrument state into register 9
Error If the registers are not located from 1 to 10, it prompts "-222, "Data out of range"".

message:

Reset state: No

*SRE

Function: Query or set the service request register 0 for disable, and 1 for enable.

Query: *SRE?

Setting: *SRE <NRf>

NRf denotes the numerical value, an integer between 0 and 255. See Table 3.3 for bit mapping.

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

*SRE 188 Set bits 2, 3, 4, 5 and 7 respectively (4+8+16+32+128).

Error message: No

Reset state: No

Table 3.3 Service request register bit mapping

Bit	Value	Description
0	1	Unused
1	2	Device information
2	4	Error/event queue
3	8	Question state
4	16	Information reception
5	32	Event state bit
6	64	Must be 0
7	128	Operation state

*STB?

Function: Query the state word.

Query: *STB?

Setting: Not supported

Example: *STB?

Error message: No

Reset state: No

Bit mapping is as follows:

Table 3.4 State word

Bit	Value	Description
0	1	Unused
1	2	Device information
2	4	Error/event queue
3	8	Question state
4	16	Information reception
5	32	Event state bit
6	64	Service request
7	128	Operation state

*TRG

Function: Trigger all channels waiting for trigger. The function of *TRG can be changed with *DDT.
Query: Not supported
Setting: *TRG
Example: *TRG
Error message: If the trigger source is not BUS, it prompts "-211, "Trigger ignored"".
Reset state: No

*TST?

Function: Execute self-test with longer self-test time.
Query: *TST?
Setting: Not supported
Example: *TST? Returning 0 means pass, and 1 means fail.
Error message: No
Reset state: No

*WAI

Function: Place 87237 in a wait state until either of the following is satisfied:
All waiting operations have been completed
Device clear command received
Restart.
Query: Not supported
Setting: *WAI

Example: *WAI
Error message: No
Reset state: No

3.3 Instrument subsystem command

This section details the subsystem commands of the 87237.

- Calculation subsystem command (CALCulate)....50
- Calibration sub-system command (CALibration)....58
- Measurement subsystem command (FETCh/READ/MEASure)....61
- Display subsystem command (DISPlay)....70
- Format subsystem command (FORMat)....71
- Memory subsystem command (MEMory/MMEMory)....72
- Sensor subsystem command (SENSe)....80
- State subsystem command (STATus)....92
- System subsystem command (SYSTem)....138
- Trace subsystem command (TRACe)....142
- Trigger subsystem command(INITiate/ TRIGger)....142
- Unit subsystem command (UNIT)....143
- Service subsystem command (SERVice)....149

3.3.1 Calculation subsystem command (CALCulate)

The calculation (CALCulate) subsystem is used for subsequent data processing and shares four independent calculation function blocks that correspond to measurements as follows:

CALC1 (Measurement 1)
CALC2 (Measurement 2)
CALC3 (Measurement 3)
CALC4 (Measurement 4)

Figure 3.1 Correspondence between CALC and measurement

Sensor (SENSe) subsystems can be used as inputs to the calculation subsystem (FEED), and the calculation diagram is as follows:

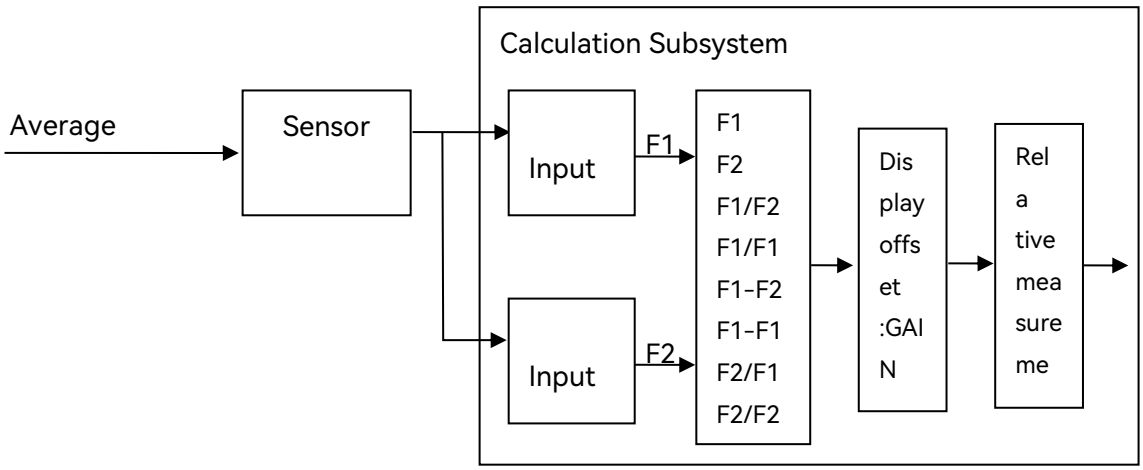


Figure 3.2 Calculation diagram

The display offset and relative measurement are calculated when the switch is turned on.

Commands include:

- :CALCulate[1]|2|3|4:FEED[1]|2.....51
- :CALCulate[1]|2-4:GAIN[:MAGNitude].....52
- :CALCulate[1]|2|3|4:GAIN:STATe.....52
- :CALCulate[1]|2|3|4:LIMit:CLEar:AUTO.....53
- :CALCulate[1]|2|3|4:LIMit:CLEar[:IMMediate].....53
- :CALCulate[1]|2|3|4:LIMit:FAIL?.....54
- :CALCulate[1]|2|3|4:LIMit:FCOunt?.....54
- :CALCulate[1]|2|3|4:LIMit:LOWer[:DATA].....54
- :CALCulate[1]|2|3|4:LIMit:STATe.....55
- :CALCulate[1]|2|3|4:LIMit:UPPer[:DATA].....56
- :CALCulate[1]|2|3|4:MATH[:EXPRession].....57
- :CALCulate[1]|2|3|4:MATH[:EXPRession]:CATalogue?.....58
- :CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO.....58
- :CALCulate[1]|2|3|4:RELative[:MAGNitude]:VALue?.....59
- :CALCulate[1]|2|3|4:RELative:STATe.....59

:CALCulate[1]|2|3|4:FEED[1]|2

Function: Query or set the measurement type.

Meaning of the suffixes in CALC: 1 for measurement 1, 2 for measurement 2, 3 for measurement 3, and 4 for measurement 4. The same below.

Meaning of the suffixes in FEED: 1 denotes input 1, 2 denotes input 2, and so on.

Query: :CALCulate[1]|2|3|4:FEED[1]|2?

Return the measurement type; refer to the settings for details.

Setting: :CALCulate[1]|2|3|4:FEED[1]|2 <string>

<string> represents the measurement type, with parameters formatted as:

“POW:AVER”: Average power

Example: CALC2:FEED? Query the measurement type of input 1 of measurement 2.

CALC:FEED2 "POW:AVER" Set input 2 of measurement 1 to the average power.

Limit: No

Error message: No

Reset state: No

:CALCulate[1]|2|3|4:GAIN[:MAGNitude]

Function: Query or set the display offset value of the specified measurement, in DB and in the range of -100 to 100. After successful setting, the display offset for this measurement is automatically turned on.

The related command is :CALCulate[1]|2|3|4:GAIN:STATe, which is used to set or query the ON/OFF state of "Display Offset".

Query: :CALCulate[1]|2|3|4:GAIN[:MAGNitude]? [MIN|MAX]

The returned value is the display offset value in the specified measurement.

Additionally, during query, the command with MIN can be used to query the minimum value that can be set, and the command with MAX can be used to query the maximum value that can be set. The same applies below.

Setting: :CALCulate[1]|2|3|4:GAIN[:MAGNitude] <numeric data>

The form of <numeric data>: DEF, MIN, MAX, and NRf, where DEF is only used for setting.

DEF represents 0dB, MIN represents -100dB, MAX represents 100dB, NRf represents floating-point values from -100 to 100.

Example: CALC:GAIN? Query the display offset of measurement 1.

CALC2:GAIN? MIN Query the minimum value of the settable display offset of measurement 2.

CALC3:GAIN MAX Set the display offset of measurement 3 to the maximum value.

CALC4:GAIN 18 Set the display offset of measurement 4 to 18 dB.

Limit: No

Error message: No

Reset state: Set to 0 (DEF).

:CALCulate[1]|2|3|4:GAIN:STATe

Function: Query or set the ON/OFF state of the display offset in the specified measurement.

The related command is :CALCulate[1]|2|3|4:GAIN[:MAGNitude], which is used to set

or query the display offset value.

Query: :CALCulate[1]|2|3|4:GAIN:STATe?

Setting: :CALCulate[1]|2|3|4:GAIN:STATe <Boolean data>

Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

CALC:GAIN:STAT?	Query the display offset ON/OFF state of measurement 1.
CALC2:GAIN:STAT ON	Enable the display offset of measurement 2.
CALC3:GAIN:STAT 0	Disable the display offset of measurement 3.

Limit: No

Error message: No

Reset state: No

:CALCulate[1]|2|3|4:LIMit:CLEar:AUTO

Function: Control when the limit FCO (failure count) is cleared.

Query: :CALCulate[1]|2|3|4:LIMit:CLEar:AUTO?

For the ONCE state, return 1 if no measurement is not started, otherwise return 0

For the OFF state, always return 0

For the ON state, return 1 if the measurement is started, otherwise return 0

Setting: :CALCulate[1]|2|3|4:LIMit:CLEar:AUTO <Boolean data>|ONCE|2

Valid forms of <Boolean data> are: 0, OFF, 1, ON.

For "ON", the FCO is set to 0 when the following operations are performed:

Initialize with the INITiate[:IMMediate] command;

Initialize with the INITiate:CONTinuous ON command;

Measure with the MEASure? command;

Read the measurement with the READ? command.

For "OFF", FCO is not cleared.

For "ONCE" or 2, it is only cleared at the first initialization and then accrued when a limit detection failure is encountered.

Example:

CALC1:LIM:CLE:AUTO?	Query FCO clearing state of measurement 1.
CALC2:LIM:CLE:AUTO ONCE	Set the FCO for clearing measurement 2 on the first initialization.

Limit: No

Error message: No

Reset state: Set to ON.

:CALCulate[1]|2|3|4:LIMit:CLEar[:IMMediate]

Function: Clear the FCO (failure count) of the specified measurement, which can be queried by CALCulate[1]|2|3|4:LIMit:FCOunt?.

Query: Not supported
Setting: :CALCulate[1]|2|3|4:LIMit:CLEar[:IMMediate]
Example: :CALC:LIM:CLE Clear the failure count of measurement 1.
Limit: No
Error message: No
Reset state: No

:CALCulate[1]|2|3|4:LIMit:FAIL?

Function: Query whether the specified measurement exceeds the limit. 1 means Yes, 0 means No.
Query: :CALCulate[1]|2|3|4:LIMit:FAIL?
Setting: Not supported
Example: :CALC:LIM:FAIL? Query the detection state of measurement 1.
Limit: No
Error message: No
Reset state: No

:CALCulate[1]|2|3|4:LIMit:FCOunt?

Function: Query the limit detection failure count (FCO) of the specified measurement.
The FCO is cleared in the following cases:
a) Reset
b) CALCulate[1]|2|3|4:LIMit:CLEar:IMMediate
c) CALCulate[1]|2|3|4:LIMit:CLEar:AUTO ON
Query: Query: :CALCulate[1]|2|3|4:LIMit:FCOunt?
Setting: Not supported
Example: :CALC:LIM:FCO? Query the detection failure count of measurement 1.
Limit: No
Error message: No
Reset state: No

:CALCulate[1]|2|3|4:LIMit:LOWer[:DATA]

Function: Query or set the lower limit of the specified measurement limit.
Query: :CALCulate[1]|2|3|4:LIMit:LOWer[:DATA]? [MIN|MAX]
Setting: :CALCulate[1]|2|3|4:LIMit:LOWer[:DATA] <numeric data>
The form of <numeric data>: DEF, MIN, MAX, and NRf, where DEF is only used for

setting.

The values of DEF, MIN, and MAX are shown in Table 3.6; the value of NRf is a floating-point number, which is actually unit-dependent. See the example for details.

If the currently set lower limit value is more than or equal to the upper limit value, the upper limit value is automatically adjusted depending on the unit, as shown in Table 3.7.

Related programmed control commands: CALCulate[1]|2|3|4:MATH[:EXPRESSION] is used to set the measurement expression for measurements 1/2/3/4;

CALCulate[1]|2|3|4:FEED[1]|2 is used to set the measurement type for measurements 1/2/3/4;

CALCulate[1]|2|3|4:RELative:STATe is used to set the relative measurement ON/OFF state for measurements 1/2/3/4;

:UNIT[1]|2|3|4:POWer is used to set units of measurements 1/2/3/4.

Example:

CALC:LIM:LOW?	Query the lower limit of measurement 1.
CALC4:LIM:LOW 0.2	According to the display unit of measurement, set the lower limit of measurement 4 as: 0.2dBm at dBm 200mW at W 0.2dB at dB 0.2% at %.

Limit: No

Error message: No

Reset state: All measurements are set to -90dBm or -90dB.

Table 3.5 Unit of measurement

Measurement Expression	Measurement type	CALC:REL:STAT OFF		CALC:REL:STAT ON	
		Linear	Logarithm	Linear	Logarithm
Single channel	Peak, average	Watt	dBm	%	dB
	Peak-to-average ratio	%	dB	%	dB
Ratio	Average, peak, peak-to-average ratio	%	dB	%	dB
Delta	Peak, average	Watt	dBm	%	dB
	Peak-to-average ratio	%	dB	%	dB

Table 3.6 Limit range

Parameter	Watt	dBm	%	dB
DEF	1pW	-90	100p%	-120
MIN	1aW	-150	100a%	-180
MAX	1XW	210	100X%	180

Table 3.7 Limit range adjustment

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]|2|3|4:LIMit:STATe

Function: Query or set the ON/OFF state of the specified measurement limit detection.
Query: :CALCulate[1]|2|3|4:LIMit:STATe?
Setting: :CALCulate[1]|2|3|4:LIMit:STATe <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON
Example: CALC:LIM:STAT? Query the limit detection ON/OFF state of measurement 1.
CALC2:LIM:STAT ON Enable the limit detection of measurement 2.
CALC3:LIM:STAT 0 Disable the limit detection of measurement 3.
Limit: When the [SENSe[1]:]MRATe command is set to FAST mode, the limit measurement is disabled, but the setting is allowed, and the setting takes effect after exiting this measurement mode.
Error message: No
Reset state: OFF: Disable the measurement limit detection

:CALCulate[1]|2|3|4:LIMit:UPPer[:DATA]

Function: Query or set the upper limit of the specified measurement limit. Refer to CALCulate[1]|2|3|4:LIMit:LOWer[:DATA]
Query: :CALCulate[1]|2|3|4:LIMit:UPPer[:DATA]? [MIN|MAX]
Setting: :CALCulate[1]|2|3|4:LIMit:UPPer[:DATA] <numeric data>
The form of <numeric data>: DEF, MIN, MAX, and NRf, where DEF is only used for setting.
The values of DEF, MIN, and MAX are shown in Table 3.8; the value of NRf is a floating-point number, which is actually unit-dependent. See the example for details.
If the currently set upper limit value is less than or equal to the lower limit value, the

lower limit value is automatically adjusted depending on the unit, as shown in Table 3.9.

Related programmed control commands: CALCulate[1]|2|3|4:MATH[:EXPRESSION] is used to set the measurement expression for measurements 1/2/3/4;

CALCulate[1]|2|3|4:FEED[1]|2 is used to set the measurement type for measurements 1/2/3/4;

CALCulate[1]|2|3|4:RELative:STATe is used to set the relative measurement ON/OFF state for measurements 1/2/3/4;

:UNIT[1]|2|3|4:POWer is used to set units of measurements 1/2/3/4.

Example: CALC:LIM:UPP? Query the lower limit of measurement 1.

CALC4:LIM:UPP 8 According to the display unit of measurement, set the upper limit of measurement 4 as:
8dBm at dBm
8W at W
8dB at dB
8% at %.

Limit: No

Error message: No

Reset state: All measurements are set to 90dBm or 90dB.

Table 3.8 Limit range

Parameter	Watt	dBm	%	dB
DEF	1MW	90	100M%	60
MIN	1aW	-150	100a%	-180
MAX	1XW	210	100X%	180

Table 3.9 Limit range adjustment

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

:CALCulate[1]|2|3|4:MATH[:EXPRession]

Function: Query or set the expression of the specified measurement: single channel, difference and ratio

Query: :CALCulate[1]|2|3|4:MATH[:EXPRession]?

Setting: :CALCulate[1]|2|3|4:MATH[:EXPRession] <string>

The form of string is as follows:

"(SENS1)", "(SENS1-SENS1)", "(SENS1/SENS1)"

Example: CALC1:MATH? Query the measurement expression of measurement 1.
CALC2:MATH "(SENS1/SENS1)" Set the expression of measurement 2 to the channel A/A ratio measurement.

Limit: No

Error message: No

Reset state: For the 87237, all measurements are set to channel A ("(SENS1)").

:CALCulate[1]|2|3|4:MATH[:EXPRession]:CATalogue?

Function: Enumerate all measurement expressions, separated by commas.
For a single-channel power meter, the string is:
"(SENS1)", "(SENS1-SENS1)", "(SENS1/SENS1)"
For a dual-channel power meter, the string is:
"(SENS1)", "(SENS2)", "(SENS1/SENS2)", "(SENS2/SENS1)",
"(SENS1-SENS2)", "(SENS2-SENS1)""(SENS1-SENS1)",
"(SENS2-SENS2)", "(SENS1/SENS1)", "(SENS2/SENS2)"

Query: :CALCulate[1]|2|3|4:MATH[:EXPRession]:CATalogue?

Setting: Not supported

Example: CALC:MATH:CAT? List all defined mathematical expressions.

Limit: No

Error message: No

Reset state: No

:CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO

Function: Set the reference value for relative measurement. In the CALCulate block, the relative value is used for the measured signal only after any mathematical computation and display offset are completed. This value shall be set to ONCE to set the reference value for relative measurement. After setting the reference value, the command returns OFF.
Additionally, when the parameter is set to ONCE, it will automatically enable relative measurement. This indicates that the programmed control command CALCulate[1]|2|3|4:RELative:STATe will be set to ON.
0|OFF does not perform any operation.
1|ON is invalid, 87237 returns an error: invalid parameter.
2|ONCE, valid parameter. Refer to the above for meaning.

Query: :CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO?

Setting: :CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO ONCE

Example: CALC:REL:AUTO? Always return 0.

	CALC:REL:AUTO ONCE	Set the reference value for the relative measurement of measurement 1.

Limit: Parameters cannot be set to 1 or ON.
Error message: If parameter 1 or ON is selected, it prompts "-224, "Illegal parameter value"".
Reset state: No

:CALCulate[1]|2|3|4:RELative[:MAGNitude]:VALue?

Function: Query the reference value for relative measurement.
Query: :CALCulate[1]|2|3|4:RELative[:MAGNitude]:VALue?
Setting: Not supported

Example:	CALC:REL:VAL?	Query the reference value for the relative measurement of measurement 1.

Limit: No
Error message: No
Reset state: No

:CALCulate[1]|2|3|4:RELative:STATe

Function: Query or set the relative measurement ON/OFF state.
When turned on, the relative measurement value set by :CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO is applied to the measurement.
Query: :CALCulate[1]|2|3|4:RELative:STATe?
Setting: :CALCulate[1]|2|3|4:RELative:STATe<Boolean data>

Example:	CALC:REL:STAT?	Return 1 when the relative measurement of measurement 1 is ON, otherwise return 0.
	CALC2:REL:STAT ON	Turn on the relative measurement state of measurement 2.

Limit: No
Error message: No
Reset state: OFF: Disable relative measurement.

3.3.2Calibration sub-system command (CALibration)

The CALibration subsystem is used to control the automatic zero bias setting and linearity adjustment of the 87237. If there is no RF signal loaded to the 87237, the zeroing setting can be done at any time.

The numeric suffix in CALibration command indicates the measurement channel.

CALibration1 and CALibration2 indicate channel A and channel B respectively (the 87237 series only supports channel A.)

Commands include:

- :CALibration[1][:ALL]……60
- :CALibration[1]:AUTO……60
- :CALibration[1]:ZERO:AUTO……61

:CALibration[1][:ALL]

Function: Zeroing/calibration of the 87237. 1 indicates channel A and can be omitted.
Zeroing and calibration are required under the following conditions:
Temperature change exceeds 5℃
After the connection is established with the device under test
Restart
After powering on the device, perform zeroing and calibration every 24 hours during operation.
every 24 hours. For example, measuring a power 20 dB higher than the minimum power of the 87237.

Query: Not supported

Setting: :CALibration[1][:ALL]

Example: CAL Zeroing and calibration

Limit: No

Error message: If there is an error in zeroing, it prompts "-231, "Data questionable;cal error"".
If there is an error in calibration, it prompts "-231, "Data questionable;zero error"".

Reset state: No

:CALibration[1]:AUTO

Function: Automatic calibration of 87237. 1 indicates channel A and can be omitted.

Query: :CALibration[1]:AUTO?

Setting: :CALibration[1]:AUTO <Boolean data>|ONCE|2
Valid forms of <Boolean data> are: 0, OFF, 1, ON.
For ON or 1, parameters are retained, and 87237 setting is invalid.
For 0FF or 0, auto-calibration is disabled
For ONCE or 2: Initiate a calibration.

Example: CAL1:AUTO ONCE Calibration of the 87237

Limit: No

Error message: If there is an error in calibration, it prompts "-231, "Data questionable; cal error"";

Reset state: No

:CALibration[1]:ZERO:AUTO

Function: Auto zeroing of 87237. 1 indicates channel A and can be omitted.
Query: :CALibration[1]:ZERO:AUTO?
Setting: :CALibration[1]:ZERO:AUTO <Boolean data>|ONCE|2
Valid forms of <Boolean data> are: 0, OFF, 1, ON.
For ON or 1, parameters are retained, and 87237 setting is invalid.
For 0FF or 0, auto-zero is disabled.
For ONCE or 2: Initiate a zeroing.
Example: CAL:ZERO:AUTO ONCE Zero channel A of the 87237
Limit: No
Error message: If there is an error in zeroing, it prompts "-231, "Data questionable;zero error"".
Reset state: ON.

3.3.3 Measurement subsystem command (FETCh/READ/MEASure)

Generally, unless otherwise specified, the linear power is in W, the logarithmic power is in dBm, the linear power ratio is in %, the logarithmic power ratio is in dB, and the time is in s.

If the returned value is invalid, NAN (9.91E37) is returned, as defined in IEEE 754.

The differences between the FETCh command, MEASure command, and READ command are as follows:

The FETCh command is to query the displayed value of a given measurement. The execution speed of the FETCh command is not related to the power level and generally returns immediately. The FETCh command may return intermediate results in the power measurement process rather than the actual power value.

The MEASure command is used to initiate a single measurement and return measurement results only after completion. The measurement duration is proportional to the average times of measurement and aperture time. Generally, lower power levels require higher auto-average count and longer MEASure command execution time. For example, when you are measuring signals below -55dBm, the auto-average count is set to 128 with a default aperture time of 50ms. This typically requires at least 6.4 seconds (128*50ms) to complete a single measurement.

The READ command completes a new measurement in the stop state, and INIT:CONT must be set to OFF. Additionally, the time required for a single READ command measurement aligns with the description provided for the MEASure command.

Note: If the trigger source is INT or EXT, after entering a new INIT operation, if the trigger conditions are not met and the trigger signal has not been waited for, the effect is that the device will be hung and the programmed command will not respond. Therefore, when using the measurement subsystem command, it is important to ensure

that the device captures the trigger signal.

Commands include:

- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]?.....62
- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?.....63
- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?.....63
- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?.....63
- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?.....64
- :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?.....64
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]?.....65
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?.....66
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?.....66
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?.....66
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?.....67
- :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RELative? 67
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]?.....68
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?.....68
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?.....69
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?.....69
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?.....69
- :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?.....70

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]?

Function: Set the specified measurement as absolute power measurement, turn off relative measurement, and return to the measurement value displayed in the current measurement. The measurement unit is specified by UNIT[1]|2|3|4:POWer. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Meaning of the suffixes in FETC: 1 for measurement 1, 2 for measurement 2, 3 for measurement 3, and 4 for measurement 4. The same below.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]?

Setting: Not supported

Example: FETC2? Set measurement 2 to the measured absolute power value.

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Function: Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Setting: Not supported

Example: FETC3:DIFF? Set measurement 3 to differential power measurement (channel A - channel A), turn off relative measurement, and return the power measurement value.

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Function: Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Setting: Not supported

Example: FETC3:DIFF:REL? Set measurement 3 to differential power measurement (channel A - channel A), turn on relative measurement, and return the power measurement value.

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

3.3 Instrument subsystem command

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Function: Set the power measurement mode of the specified measurement as ratio measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Setting: Not supported

Example: FETC4:RAT? Set measurement 4 to ratio power measurement (channel A divided by channel A), turn off relative measurement, and return the measurement value.

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Function: Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Setting: Not supported

Example: FETC4:RAT:REL? Set measurement 4 to ratio power measurement (channel A divided by channel A), turn on relative measurement, and return the measurement value

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Function: Set the specified measurement to absolute power measurement and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. This command does not wait for a measurement to be completed before returning. To get the exact measurement value, use the MEAS command.

Query: :FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Setting: Not supported

Example: FETC2:REL? Query the relative power measurement value of measurement 2.

Limit: No.

Error message: If the last measurement is invalid, it prompts -230, "Data corrupt or stale". Running the INIT command will ensure that the measurement is valid, and changes in measurement parameters (e.g., frequency, channel offset) will cause the measurement to be invalid.

Reset state: No.

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]?

Function: Set the specified measurement as absolute power measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed. When composing a test system, please use the MEASure command.

Meaning of the suffixes in MEAS: 1 for measurement 1, 2 for measurement 2, 3 for measurement 3, and 4 for measurement 4. The same below.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]?

Setting: Not supported

Example: MEAS2? Start one measurement and return to measurement 2 for the absolute power measurement after the measurement is completed.

Limit: No

Error message: No

Reset state: No

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Function: Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed When composing a test system, please use the MEASure command.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Setting: Not supported

Example: MEAS3:DIFF? Set measurement 3 to differential power measurement (channel A – channel A), turn off relative measurement, start one measurement, and return the power measurement value when the measurement is complete.

Limit: No

Error message: No

Reset state: No

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Function: Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed When composing a test system, please use the MEASure command.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Setting: Not supported

Example: MEAS3:DIFF:REL? Set measurement 3 to differential power measurement (channel A – channel A) with a resolution of 3, turn on relative measurement, start one measurement, and return the power measurement value when the measurement is complete.

Limit: No

Error message: No

Reset state: No

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Function: Set the power measurement mode of the specified measurement as ratio measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed. When composing a test system, please use the MEASure command.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Setting: Not supported

Example: MEAS4:RAT? Set measurement 4 to ratio power measurement (channel A divided by channel A), turn off relative measurement, start one measurement, and return the power measurement Ratio when the measurement is complete.

Limit: No

Error message: No

Reset state: No

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Function: Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed. When composing a test system, please use the MEASure command.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Setting: Not supported

Example: MEAS4:RAT:REL? Set measurement 4 to ratio power measurement (channel A divided by channel A) with a resolution of 3, turn on relative measurement, start one measurement, and return the power measurement Ratio when the measurement is complete.

Limit: No

Error message: No

Reset state: No

:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Function: Set the specified measurement to absolute power measurement and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer. Start a measurement and return the measurement value when the measurement is completed. And the corresponding FETCh command returns the current measurement value, not when the measurement is completed. When composing a test system, please use the MEASure command.

Query: :MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Setting: Not supported

Example: MEAS2:REL? Start a measurement and return the relative power measurement of measurement 2 when the measurement is completed.

Limit: No

Error message: No

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]?

Function: Set the specified measurement as absolute power measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer.
Meaning of the suffixes in READ: 1 for measurement 1, 2 for measurement 2, 3 for measurement 3, and 4 for measurement 4. The same below.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]?

Setting: Not supported

Example: READ2? Query that measurement 2 is the absolute power measurement value.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Function: Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?

Setting: Not supported

Example: READ3:DIFF? Set measurement 3 to differential power measurement (channel A - channel A), turn off relative measurement, and return the power measurement value.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Function: Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value. The measurement unit is specifiec by UNIT[1]|2|3|4:POWer:RATio.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?

Setting: Not supported

Example: READ3:DIFF:REL? Set measurement 3 to differential power measurement (channel A - channel A), turn on relative measurement, and return the power measurement value.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Function: Set the power measurement mode of the specified measurement as ratic measurement, turn off relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?

Setting: Not supported

Example: READ4:RAT? Set measurement 4 to ratio power measurement (channel A divided by channel A), turn off relative measurement, and return the measured power value.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Function: Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?

Setting: Not supported

Example: READ4:RAT:REL? Set measurement 4 to ratio power measurement (channel A divided by channel A), turn on relative measurement, and return the measured power value.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Function: Set the specified measurement to absolute power measurement and return the measurement value. The measurement unit is specified by UNIT[1]|2|3|4:POWer:RATio.

Query: :READ[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?

Setting: Not supported

Example: READ2:REL? Query the relative power measurement value of measurement 2.

Limit: No

Error message: If INIT:CONT is set to ON, it prompts -213, "Init ignored".

Reset state: No

3.3.4 Display subsystem command (DISPlay)

The DISPlay subsystem is used to control the display of text, graphics and traces.

Commands include:

- :DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution.....71

:DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution

Function: Query or set the display resolution of the specified measurement. WINDow1 indicates the upper window, which can be omitted, and WINDow2 indicates the lower window; NUMeric1 indicates the upper measurement, which can be omitted and NUMeric2 indicates the lower measurement.

Query: :DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution?

Setting: :DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution <NRf>

NRf valid values are 1, 2, 3, 4.

Example:

DISP:WIND2:NUM2:RES?	Query the display resolution of measurement 4.
DISP:RES 3	Set the display resolution of measurement 1 to 3, i.e., display 3 valid digits for linear display and 2 decimal places (0.01) for logarithmic display.

Limit: No

Error message: No

Reset state: Resolution is 3

3.3.5 Format subsystem command (FORMat)

The format subsystem is used to set the transmission format of the numeric data. This format system is only used for the following query commands:

FETCh?

MEASure?

READ?

- :FORMat[:READings]:BORDer.....71
- :FORMat[:READings][:DATA].....72

:FORMat[:READings]:BORDer

Function: Query or set the transmission order of binary data: normal or byte exchange. Valid only when FORMat[:READings][:DATA] is set to REAL.

Query: :FORMat[:READings]:BORDer?

Setting: :FORMat[:READings]:BORDer <character data>

Valid values for <character data> are: 0 or NORMal, 1 or SWAPped

Example:

FORM:BORD?	Query the transmission order.

	FORM:BORD SWAP	Set the transmission order to byte-swap.
Limit:	No	
Error message:	No	
Reset state:	Set to NORMal	

:FORMat[:READings][:DATA]

Function:	Query or set the data transfer format: ASCii and REAL (real numbers)
Query:	:FORMat[:READings][:DATA]?
Setting:	:FORMat[:READings][:DATA] <character data> Valid values for <character data> are: 0 or ASCii, 1 or REAL ASCii: Numerical data is transferred in the form of <NRf>. REAL: Numerical data is transferred in IEEE 754 64-bit floating point format and is 8 bytes per data.

Example:	FORM?	Query the data transmission format.
	FORM REAL	Set the data transmission format to REAL.
Limit:	No	
Error message:	No	
Reset state:	Set to ASCii	

3.3.6 Memory subsystem command (MEMory/MMEMory)

- :MEMory:CATalog[:ALL]? ... 73
- :MEMory:CATalog:STATe? ... 73
- :MEMory:CATalog:TABLe? ... 74
- :MEMory:CLEar[:NAME] ... 74
- :MEMory:CLEar:TABLe[1]|2|3|4|5|6|7|8|9|10 ... 75
- :MEMory:FREE[:ALL]? ... 75
- :MEMory:FREE:STATe? ... 75
- :MEMory:FREE:TABLe? ... 76
- :MEMory:NSTates? ... 76
- :MEMory:STATe:CATalog? ... 76
- :MEMory:STATe:DEFine ... 77
- :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine ... 77
- :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency ... 77
- :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency:POINts? ... 78
- :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude] ... 78
- :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]:POINts? ... 79
- :MEMory:TABLe:MOVE ... 79

- :MEMory:TABLe:SELect……80

:MEMory:CATalog[:ALL]?

Function: Enumerate the user configurations in the 87237, including storage call configuration, frequency response offset table (FDO), etc.
The data format is: <value 1>, <value 2> {, <string>}
Where <value 1> indicates the length of bytes used by the user configuration and <value 2> indicates the length of unused bytes.
The form of each <string> is as follows:
<string i>, <type>, <length>
<string i> indicates the name of the user configuration.
<type> indicates the type of user configuration, TABL indicates frequency response offset table, and STAT indicates storage call configuration.
<length> indicates the length in bytes of this configuration item.

Query: :MEMory:CATalog[:ALL]?

Setting: Not supported

Example: MEM:CAT? Enumerate all user configurations in the 87237.

Limit: No

Error message: No

Reset state: No

:MEMory:CATalog:STATe?

Function: Enumerate the storage call configuration in the 87237.
The data format is: <value 1>, <value 2> {, <string>}
Where <value 1> indicates the length of bytes used by the user configuration and <value 2> indicates the length of unused bytes.
The form of each <string> is as follows:
<string i>, <type>, <length>
<string i> indicates the name of the user configuration.
<type> indicates the type of user configuration, and STAT indicates storage call configuration.
<length> indicates the length in bytes of this configuration item.

Query: :MEMory:CATalog:STATe?

Setting: Not supported

Example: MEM:CAT:STAT? Enumerate the storage call configuration in the 87237.

Limit: No

Error message: No

Reset No

state:

:MEMory:CATalog:TABLe?

Function: Enumerate the frequency response offset table in the 87237.
The data format is: <value 1>, <value 2> {, <string>}
Where <value 1> indicates the length of bytes used by the user configuration anc <value 2> indicates the length of unused bytes.
The form of each <string> is as follows:
<string i>, <type>, <length>
<string i> indicates the name of the user configuration.
<type> indicates the type of user configuration, and TABL indicates frequency response offset table.
<length> indicates the length in bytes of this configuration item.

Query: :MEMory:CATalog:TABLe?

Setting: Not supported

Example: MEM:CAT:TABL? Enumerate the frequency response offset table in the 87237.

Limit: No

Error message: No

Reset state: No

:MEMory:CLEar[:NAME]

Function: It is used to clear the frequency response offset table or storage call table specifiec in the power.

Query: Not supported

Setting: :MEMory:CLEar[:NAME] <string>
<string> indicates the name of frequency response offset table or storage call table

Example: MEM:CLE "fdo0" Clear the fdo0 frequency response offset table.

Limit: 1) If the specified frequency response offset table is enabled, it is automatically turned off.
2) Clear the frequency response offset table if the specified name can be found ir the frequency response offset list;
3) Clear the state table if the specified name can be found in the storage call states list.
4) If a frequency response offset table exists with the same name as a storage cal table (e.g., both named “State1”), the MEM:CLE “State1” command will be called anc both the corresponding frequency response offset table and storage call table wil be cleared.

Error If the specified name does not exist, it prompts "-224, "Illegal parameter value""

message:
Reset state: No

:MEMory:CLEar:TABLe[1]|2|3|4|5|6|7|8|9|10

Function: Clear the specified frequency response offset table.
Meaning of the suffixes in TABL: 1–10 denote frequency response offset tables 1 through 10 (the same below).
Query: Not supported
Setting: :MEMory:CLEar:TABLe[1]|2|3|4|5|6|7|8|9|10
Example: MEM:CLE:TABL5 Clear the 5th frequency response offset table. (10 in total)
Limit: No
Error message: No
Reset state: No

:MEMory:FREE[:ALL]?

Function: Query the total number of bytes unused in the user configuration space and the number of bytes used.
The return string is of the form: <unused bytes>, <used bytes>
Query: :MEMory:FREE[:ALL]?
Setting: Not supported
Example: MEM:FREE? Return total number of bytes unused in the user configuration space and the number of bytes.
Limit: No
Error message: No
Reset state: No

:MEMory:FREE:STATe?

Function: Query the total number of bytes unused in the storage call space and the number of bytes used.
The return string is of the form: <unused bytes>, <used bytes>
Query: :MEMory:FREE:STATe?
Setting: Not supported
Example: MEM:FREE:STAT? Return total number of bytes unused in the storage call space and the number of bytes.
Limit: No

Error message: No

Reset state: No

:MEMory:FREE:TABLe?

Function: Query the total number of unused bytes and the number of bytes used in the frequency response offset table space.
The return string is of the form: <unused bytes>, <used bytes>

Query: :MEMory:FREE:TABLe?

Setting: Not supported

Example: MEM:FREE:TABL? Return total number of unused bytes and the number of bytes used in the frequency response offset table.

Limit: No

Error message: No

Reset state: No

:MEMory:NSTates?

Function: Query the number of storage call states, and always return 10

Query: :MEMory:NSTates?

Setting: Not supported

Example: MEM:NST? Return the number of storage call states: 10

Limit: No

Error message: No

Reset state: No

:MEMory:STATe:CATalog?

Function: Enumerate the names of all storage call states.

Query: :MEMory:STATe:CATalog?

Setting: Not supported

Example: MEM:STAT:CAT? Return the names of all storage call states of the 87237

Limit: No

Error message: No

Reset state: No

:MEMory:STATe:DEFine

Function: Query the register number corresponding to a storage call state name, or set the name of a storage call state register.

Query: :MEMory:STATe:DEFine? <string>

The string is the name of the instrument state contained in quotation marks, such as "State1".

Returned value is the state register number corresponding to the specified name.

Setting: :MEMory:STATe:DEFine <string>, <numeric data>

The definition of string is the same as that of query format;

Valid values for numeric data are: DEF, MIN, MAX, <NRf>, which are used for setting only.

<NRf> is an integer ranging from 1 to 10.

Example:

MEM:STAT:DEF? "State1"	Query the state register number corresponding to the "State1".
MEM:STAT:DEF "MyState",8	Name the state register 8 as "MyState"

Limit: No

Error message: If the character string of the instrument state name does not exist during query, it prompts "-224, "Illegal parameter value"".

If the length of the first parameter name is longer than 12 characters during setting it prompts "-257, "File name error"”

Reset state: No

:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine

Function: Query or set the name in the specified frequency response offset table.

Query: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine?

Setting: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine <string>

Example:

MEM:TABL6:DEF?	Query the name of the 6th frequency response offset table.
MEM:TABL8:DEF "fdo0"	Name the 8th frequency response offset table as fdo0.

Limit: The specified name shall be unique among existing table names, with no more than 12 characters.

Error message: If the specified name already exists, it prompts "-257, "File name error"".

If the length of the specified name is longer than 12 characters, it prompts "-257 "File name error"”

Reset state: No

:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency

Function: Query or set the frequency list in the specified frequency response offset table.

When setting the frequency list, the previous frequency list will be cleared and the frequency list must be sorted in ascending order.

The relevant command is:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]

Query: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency?

Setting: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency <numeric data 1>{, <numeric data n>}

It is necessary to ensure that this frequency list covers the frequency range of the 87237, and if the frequency of the signal under test is outside the range of this list the value at the lowest or highest frequency will be used.

The maximum number of frequency points is 4000.

The effective units of frequency are: Hz, kHz, MHz, and GHz

Example: MEM:TABL6:FREQ? Query the frequency list of the 6th frequency response offset table.

MEM:TABL6:FREQ 50MHz, 40GHz Set the frequency of the 6th frequency response offset table to 50MHz and 40GHz.

Limit: Frequency lists shall be sorted in ascending order and contain no more than 4000 points.

Error message: If the number of points in the frequency list exceeds 4000, it prompts "-108, "Parameter not allowed"".

If the frequency list is not in increasing order, it prompts "-220, "Parameter error"”

Reset state: No

:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency:POINts?

Function: Query the frequency points of the frequency response offset table.

Query: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency:POINts?

Setting: Not supported

Example: MEM:TABL6:FREQ:POIN? Query the frequency points of the 6th frequency response offset table.

Limit: No

Error message: No

Reset state: No

:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]

Function: Query or set the offset list in the specified frequency response offset table.

When you are setting the offset list, the previous offset list will be cleared,

The related command is:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency

Query: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]?

Setting: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude] <numeric data 1>{, <numeric data n>}

The maximum number of offset points is 4000.

Numeric data is expressed in PCT.

The range of the offset is from (1.0e-009) × 100% to (1.0e+009) × 100%, corresponding to a range of -90dB to 90dB.

Example:	MEM:TABL6:GAIN?	Query the offset list of the 6th frequency response offset table.
	MEM:TABL6:GAIN 98, 102	Set the offset list of the 6th frequency response offset table to 98% and 102%.

Limit: The offset list shall not contain 4000 points or more.

Error message: No

Reset state: No

:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]:POINts?

Function: Query the number of offset points of the frequency response offset table.

Query: :MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]:POINts?

Setting: Not supported

Example:	MEM:TABL6:GAIN:POIN?	Query the number of offset points of the 6th frequency response offset table.

Limit: No

Error message: No

Reset state: No

:MEMory:TABLe:MOVE

Function: Rename the specified frequency response offset table. This command requires prior knowledge of the name of the frequency response offset table to be modified.

Related commands MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine

It is recommended to use MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine.

Query: Not supported

Setting: :MEMory:TABLe:MOVE < string 1>, < string 2>

<string 1> indicates the name of the frequency response offset table that needs to be modified

<string 2> indicates the name of the modified frequency response offset table

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

Limit: The table corresponding to the first parameter shall exist. The second parameter shall be unique among other table names, with no more than 12 characters.

Error message: If the frequency response offset table specified by the first parameter does not exist it prompts "-256, "File name not found""

If the frequency response offset table specified by the second parameter already exists, it prompts "-257, "File name error"”

If the length of the second parameter is longer than 12 characters, it prompts "-257 "File name error""

Reset state: No

:MEMory:TABLe:SELect

Function: Query or set the current frequency response offset table.

Query: :MEMory:TABLe:SELect?

Setting: :MEMory:TABLe:SELect <string>

Example:

MEM:TABL:SEL?	Return the name of the current frequency response offset table.
MEM:TABL:SEL "fdo0"	Set the current frequency response offset table to fdo0.

Limit: The specified frequency response offset table must exist.

Error message: If the frequency response offset table specified by the parameter does not exist, it prompts "-224, "Illegal parameter value"".

Reset state: No

3.3.7 Sensor subsystem command (SENSe)

The SENSe subsystem is used to configure various setting parameters for the power sensor. Numeric suffixes in SENSe commands denote measurement channels, where SENSe1 and SENSe2 represent channel A and channel B, respectively. It is particularly worth noting that, to maintain the inheritance of programmed control commands of the microwave power meter produced by Ceyear, the 87237 series, as a single sensor, only has channel A. However, to achieve forward compatibility with the programmed control commands of the 2438 series microwave power meter, the 87237 series also supports receiving SENSe2-related commands and the functionality of these commands is the same as that of SENSe1. Certain channel-related commands such as INITiate, TRIGer, SEQuence, SENSor, etc., also adopt this setting, which will not be repeated hereafter.

- [:SENSe[1]]:AVERage:COUNt 81
- [:SENSe[1]]:AVERage:COUNt:AUTO 82
- [:SENSe[1]]:AVERage:RESet 82
- [:SENSe[1]]:AVERage:SDETect 82
- [:SENSe[1]]:AVERage[:STATe] 83
- [:SENSe[1]]:BUFFer:COUNt 83
- [:SENSe[1]]:CORRection:CSET2[:SELect] 84
- [:SENSe[1]]:CORRection:CSET2:STATe 84
- [:SENSe[1]]:CORRection:DCYCle[:INPut][:MAGNitude] 84
- [:SENSe[1]]:CORRection:DCYCle[:INPut]:STATe 85

- [:SENSe[1]]:CORRection:FDOFfset[:INPut][:MAGNitude]?.....84
- [:SENSe[1]]:CORRection:GAIN[1]|2|3|4[:INPut][:MAGNitude].....86
- [:SENSe[1]]:CORRection:GAIN[1]|2|3|4[:INPut]:STATe.....87
- [:SENSe[1]]:DETector:FUNCtion.....87
- [:SENSe[1]]:FREQuency[:CW|FIXed].....87
- [:SENSe[1]]:FREQuency[:CW|FIXed]:STARt.....88
- [:SENSe[1]]:FREQuency[:CW|FIXed]:STEP.....88
- [:SENSe[1]]:FREQuency[:CW|FIXed]:STOP.....89
- [:SENSe[1]]:MRATe.....90
- [:SENSe[1]]:POWer:AC:RANGe.....90
- [:SENSe[1]]:POWer:AC:RANGe:AUTO.....91
- [:SENSe[1]]:SWEep:APERture.....91
- [:SENSe[1]]:SWEep:APERture:AUTO.....92

[:SENSe[1]]:AVERage:COUNt

Function: Query and set the average times of measurement
Measurement average, where consecutive measurements are averaged in order tc improve the accuracy of the measurements.
Setting measurement average will automatically perform the following commanc settings:
AVERage:COUNt:AUTO OFF
AVERage1:STATe ON

Query: [:SENSe[1]]:AVERage:COUNt? [MIN|MAX]

Setting: [:SENSe[1]]:AVERage:COUNt <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> is an integer between 1 and 262144 for average number of measurements,
DEF is 8,
MIN is 1,
MAX is 262144.

Example:

SENS1:AVER:COUN?	Query the average number of measurements.
SENS1:AVER:COUN? MAX	Query the maximum value of the settable average number of measurements.
SENS:AVER:COUN 28	Set the average number of measurements to 28.

Limit: No

Error message: No

Reset state: Average times is set to 8.

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

Function: Query or set the measurement auto-average state.
Query: [:SENSe[1]]:AVERage:COUNt:AUTO?
Setting: [:SENSe[1]]:AVERage:COUNt:AUTO <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON
Example: SENS1:AVER:COUN:AUTO? Query the automatic average state.
SENS:AVER:COUN:AUTO 1 Turn on auto average.
Limit: No
Error message: No
Reset state: Auto average is set to ON.

[:SENSe[1]]:AVERage:RESet

Function: Clear the averaging buffer and restart averaging of the 87237.
The 87237 is equipped with an internal averaging buffer for storing historical measurement data. This command is used to clear this buffer and re-store it.
Query: Not supported
Setting: [:SENSe[1]]:AVERage:RESet
Example: AVER:RES Re-average
Limit: No
Error message: No
Reset state: No

[:SENSe[1]]:AVERage:SDETect

Function: Query or set the channel step detection state.
In auto average mode, the last four averages are used to compare with the entire filter value, and the digital filter is cleared when the difference between the two averages exceeds 15%. The filter then starts storing the new measurement values The use of this function is able to shorten the filtering time when the power changes more significantly.
Query: [:SENSe[1]]:AVERage:SDETect?
Setting: [:SENSe[1]]:AVERage:SDETect <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON
Example: SENS1:AVER:SDET? Query the step detection state.
SENS:AVER:SDET 1 Turn on the step detection.
Limit: No
Error message: No

Reset state: Step detection is set to ON.

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

Function: Query or set the ON/OFF state of measurement average.

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

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

Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

SENS1:AVER?	Query the ON/OFF state of measurement average.
SENS:AVER 1	Turn on the measurement average.

Limit: No

Error message: No

Reset state: Average state is set to ON.

[:SENSe[1]]:BUFFer:COUNt

Function: Query or set the buffer size of external trigger measurement, which is only used for external trigger measurement.

Notice: This command can be set only when [:SENSe[1]]:FREQuency[:CW|FIXed]:STEP (i.e., the number of frequency steps during sweep measurements) is set to 0. Otherwise, an error will occur.

Query: [:SENSe[1]]:BUFFer:COUNt?

Setting: [:SENSe[1]]:BUFFer:COUNt <numeric data>

Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.

<NRf> ranges from 1 to 80000

DEF is 1.

MIN is 1.

MAX is 80000.

Example:

SENS:BUFF:COUN?	Query the current buffer size of external trigger measurement.
SENS:BUFF:COUN 100	Set the buffer size of external trigger measurement to 100.

Limit: The trigger source shall be “External Trigger,” and the number of frequency steps shall be “0”.

Error message: If the trigger source (TRIG:SOUR) is not "external trigger", it prompts "-221, "Settings conflict"".

If the number of frequency steps (FREQ:STEP) is not 0, it prompts "-221, "Settings conflict"".

Reset 1

state:

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

Function: Set or query the name of the frequency response offset table for the current device. The data of the frequency response offset table is shared among channels, but the ON/OFF states are separate.

Query: [:SENSe[1]]:CORRection:CSET2[:SELect]?
Return the name of the currently selected frequency response offset table.

Setting: [:SENSe[1]]:CORRection:CSET2[:SELect] <string>
<string> indicates the name of the frequency response offset table.

Example:

SENS:CORR:CSET2?	Query the currently selected frequency response offset table.
SENS:CORR:CSET2 "fdo0"	Select the frequency response offset table named "fdo0".

Limit: No

Error message: If the frequency response offset table specified by the parameter does not exist, it prompts "-256, "File name not found"".

Reset state: No

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

Function: Query or set the activation state of the frequency response offset table.
The data of the frequency response offset table is shared among channels, but the activation states are separate.

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

Setting: [:SENSe[1]]:CORRection:CSET2:STATe <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

SENS1:CORR:CSET2:STAT?	Query the activation state of the frequency response offset table.
SENS:CORR:CSET2:STAT 0	Disable the frequency response offset table.

Limit: No

Error message: If the number of frequency points and the number of amplitude offset points in the currently selected frequency response offset table are different, it prompts "-226," list length is not the same ""
If the point in the currently selected frequency response offset table is 0, it prompts "-221, "Settings conflict"".

Reset state: Do not affect the activation state of the frequency response offset table.

[:SENSe[1]]:CORRection:DCYCle[:INPut][:MAGNitude]

Function: Query or set the channel duty cycle setting value for pulse power measurement.
Pulse power measurements average any aberrations such as overshoot, ringing

etc. The measurement is a mathematical representation of the pulse power measurement and not a true measurement. The 87237 measures the average power of the pulse signal by dividing it by the duty cycle to obtain a pulse power reading.

Note: To ensure the accuracy of the measurement, the input signal must be a rectangular pulse, other pulses (such as triangle wave, linear FM pulse) will lead to inaccurate measurement.

The switching ratio of the pulse under test must be much larger than the duty cycle setting.

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

Setting: [:SENSe[1]]:CORRection:DCYCle[:INPut][:MAGNitude] <numeric data>

Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.

<NRf> ranges from 0.001 to 100, in %.

DEF is 100,

MIN is 0.001,

MAX is 100.

Example:

SENS:CORR:DCYC?	Query the current duty cycle setting value of the channel.
CORR:DCYC? MAX	Query the maximum settable duty cycle.
CORR:DCYC 50	Set the channel duty cycle to 50%.

Limit: No

Error message: No

Reset state: 100%

[:SENSe[1]]:CORRection:DCYCle[:INPut]:STATe

Function: Query or set the ON/OFF state of channel duty cycle for pulse power measurement.

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

Setting: [:SENSe[1]]:CORRection:DCYCle[:INPut]:STATe <Boolean data>

Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

SENS:CORR:DCYC:STAT?	Query the activation state of duty cycle of the channel.
CORR:DCYC:STAT 1	Enable the duty cycle correction for the channel.

Limit: No

Error message: No

Reset state: OFF

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

Function: Query the currently used frequency response offset factor. In PCT.

Query: [:SENSe[1]]:CORRection:FDOFfset[:INPut][:MAGNitude]?
The returned value is the currently used frequency response offset factor, in PCT.

Setting: Not supported

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

Limit: No

Error message: No

Reset state: No

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

Function: It is used to set or query four kinds of gains for the current device;
The equivalent form of GAIN [1] is CFACtor, which denotes the calibration factor of the 87237. The former conforms to the SCPI specification, while the latter does not GAIN [1] is not supported in this case;
GAIN2 indicates the channel offset. It is supported in this case. It can be queried or set.
The equivalent form of GAIN3 is DCYCle, which represents the duty cycle. It is supported in this case. It can be queried or set.
The equivalent form of GAIN4 is FDOFfset, which represents the frequency response offset factor. GAIN4 is supported in this case. The frequency response offset factor is only available for query.

Query: [:SENSe[1]]:CORRection:GAIN2|3[:INPut][:MAGNitude]? [MIN|MAX]
[:SENSe[1]]:CORRection:GAIN4[:INPut][:MAGNitude]?

Setting: [:SENSe[1]]:CORRection:GAIN2|3[:INPut][:MAGNitude] <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
For GAIN2, the values are as follows:
<NRf> ranges from -100 to 100 (unit: dB)
DEF is 0dB,
MIN is -100dB,
MAX is 100dB.
For GAIN3, the values are as follows:
<NRf> ranges from 1e-3 to 100, in %,
DEF is 100%,
MIN is 0.001%,
MAX is 100%.

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 3.6dB.

Limit: 87237 does not support GAIN[1]; GAIN4 supports query only.

Error message: No

Reset state: Channel offset is 0; duty cycle is 100%

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

Function: It is used to set or query the ON/OFF state of four kinds of gains for the current device;
The equivalent form of GAIN [1] is CFACtor, which denotes the calibration factor of the 87237. The former conforms to the SCPI specification, while the latter does not GAIN [1] is not supported in this case;
GAIN2 indicates the channel offset. It is supported in this case. It can be queried or set.
The equivalent form of GAIN3 is DCYCle, which represents the duty cycle. It can be queried or set.
The equivalent form of GAIN4 is FDOFfset, which represents the frequency response offset factor. GAIN4 is not supported in this case

Query: [:SENSe[1]]:CORRection:GAIN2|3[:INPut]:STATe?

Setting: [:SENSe[1]]:CORRection:GAIN2|3[:INPut]:STATe <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:	SENS:CORR:GAIN2:STAT?	Query the activation state of channel offset.
	CORR:GAIN2:STAT 1	Enable the channel offset.

Limit: 87237 does not support GAIN[1] and GAIN4.

Error message: No

Reset state: OFF

[: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>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> ranges from 0 to 1e12, in Hz,
DEF is 50MHz,
MIN is 0Hz,

MAX is 1000GHz,

Example:	FREQ?	Query the frequency.
	FREQ? MAX	Query the maximum settable frequency.
	SENS:FREQ 8GHz	Set the frequency to 8GHz.
Limit:	No	
Error message:	No	
Reset state:	50MHz.	

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

Function: Query or set the frequency for the start frequency of the external trigger buffer sweep measurement.

Associated commands:

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

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

The start frequency can be greater than or equal to the stop frequency.

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

Setting: [:SENSe[1]]:FREQuency[:CW|FIXed]:STARt <numeric data>

Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.

<NRf> ranges from 0 to 1e12, in Hz,

DEF is 50MHz,

MIN is 0Hz,

MAX is 1000GHz.

Example:	FREQ:STAR?	Query the start frequency of sweep measurement.
	FREQ:STAR? MAX	Query the maximum settable start frequency of sweep measurement.
	SENS:FREQ:STAR 8GHz	Set the start frequency of sweep measurement to 8GHz.
Limit:	No	
Error message:	No	
Reset state:	50MHz.	

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

Function: Query or set the frequency step number for the external trigger buffer sweep measurement.

Associated commands:

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

[:SENSe[1]]:FREQuency[:CW|FIXed]:STOP
[:SENSe[1]]:BUFFer:COUNt
The number of frequency steps can be 0. When it is 0, the [:SENSe[1]]:BUFFer:COUNt
command can be used to set the buffer measurement size; when it is not 0, the set value of [:SENSe[1]]:BUFFer:COUNt is invalid. The following equation can be used to calculate the number of frequency steps (Step) (Start, Stop, and Interval are the start frequency, stop frequency and frequency interval respectively. The Interval cannot be 0).

$$Step = (\text{int})\left|\frac{Start - Stop}{Interval}\right| + 1$$

Query: [:SENSe[1]]:FREQuency[:CW|FIXed]:STEP? [MIN|MAX]
Setting: [:SENSe[1]]:FREQuency[:CW|FIXed]:STEP <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> is an integer ranging from 0 to 80000,
DEF is 0,
MIN is 0,
MAX is 80000.

Example:		
	FREQ:STEP?	Query the number of frequency steps.
	FREQ:STEP? MAX	Query the maximum settable frequency steps.
	SENS:FREQ:STEP 100	Set the number of frequency steps to 100.

Limit: No
Error message: No
Reset state: 0.

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

Function: Query or set the frequency for the stop frequency of the external trigger buffer sweep measurement.
Associated commands:
[:SENSe[1]]:FREQuency[:CW|FIXed]:STARt
[:SENSe[1]]:FREQuency[:CW|FIXed]:STEP
The start frequency can be greater than or equal to the stop frequency.
Query: [:SENSe[1]]:FREQuency[:CW|FIXed]:STOP? [MIN|MAX]
Setting: [:SENSe[1]]:FREQuency[:CW|FIXed]:STOP <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> ranges from 0 to 1e12, in Hz,
DEF is 50MHz,
MIN is 0Hz,

MAX is 1000GHz.

Example:	FREQ:STOP?	Query the end frequency of sweep measurement.
	FREQ:STOP? MAX	Query the maximum settable stop frequency of sweep measurement.
	SENS:FREQ:STOP 8GHz	Set the end frequency of sweep measurement to 8GHz.

Limit: No

Error message: No

Reset state: 50MHz.

[:SENSe[1]]:MRATe

Function: Query or set the measurement speed.

Query: [:SENSe[1]]:MRATe?

Setting: [:SENSe[1]]:MRATe <character data>

The valid values of the character data are as follows:

NORMal or 0: 20 readings per second at normal speed

DOUBle or 1: 40 readings per second at doubled speed

FAST or 2: 1000 readings per second in Fast mode

Note: In the fast test mode, the following setting parameters are ignored anc processed by "OFF": average state, channel offset state, measurement offset state relative measurement state, and limit state.

Example:	MRAT?	Query the measurement speed
	SENS:MRAT FAST	Set the quick measurement

Limit: No

Error message: No

Reset state: Normal speed

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

Function: Query or set the sensor range (4 ranges total).

After setting the range with this command, auto range state switches to Manual, i.e., [SENSe[1]]:POWer:AC:RANGe:AUTO is set to OFF.

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

Setting: [:SENSe[1]]:POWer:AC:RANGe <0|1|2|3>

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

Limit: No

Error No

message:
Reset state: No

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

Function: Query or set the ON/OFF state of auto range of the sensor. 0 indicates Manual, 1 indicates Auto.
When auto range is enabled, the power sensor automatically adjusts its range based on the power signal, which is measured with the optimal range. Related commands: [:SENSe[1]]:POWer:AC:RANGe.
Query: [:SENSe[1]]:POWer:AC:RANGe:AUTO?
Setting: [:SENSe[1]]:POWer:AC:RANGe:AUTO <Boolean Data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON
Example: SENS:POW:AC:RANG:AUTO? Query the auto range ON/OFF state.
POW:AC:RANG:AUTO 0 Set auto range to Manual.
Limit: No
Error message: No
Reset state: Set to auto range.

[:SENSe[1]]:SWEep:APERture

Function: Query or set the measurement aperture or measurement time interval.
The relevant command is: [:SENSe[1]]:SWEep:APERture:AUTO
After the measurement aperture is set, the auto-measurement aperture state is set to OFF.
Query: [:SENSe[1]]:SWEep:APERture? [MIN|MAX]
Setting: [:SENSe[1]]:SWEep:APERture <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> ranges from 1e-3 to 3e-1, in seconds.
DEF is 50ms,
MIN is 1ms,
MAX is 300ms.
Example: SENS:SWE:APER? Query the current measured aperture.
SWE:APER 1e-3 Set the measurement aperture to 1ms
Limit: It can be set only when the trigger source is set to IMMediate
Error message: No
Reset state: 50ms

[:SENSe[1]]:SWEep:APERture:AUTO

Function: Query or set the auto-measurement aperture or measurement interval state.
If it is set to auto-measurement aperture, the measured aperture value will be automatically selected according to the measurement speed, and the corresponding relationship is as follows:
Normal measurement speed, with an aperture of 50ms;
Doubled measurement speed with an aperture of 25ms;
Quick measurement speed with an aperture of 1ms;

Query: [:SENSe[1]]:SWEep:APERture:AUTO

Setting: [:SENSe[1]]:SWEep:APERture:AUTO <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

SENS:SWE:APER:AUTO?	Query the current auto-measurement aperture state.
SWE:APER:AUTO 1	Set to auto-measurement aperture

Limit: No

Error message: No

Reset state: OFF

3.3.8 State subsystem command

The state subsystem command detects the state of the 87237 by monitoring the device state register, the operation state register, and the question register.

- :STATus:DEVice:CONDition?.....115
- :STATus:DEVice:ENABle.....116
- :STATus:DEVice[:EVENt]?.....116
- :STATus:DEVice:NTRansition.....116
- :STATus:DEVice:PTRansition.....117
- :STATus:OPERation:CALibrating[:SUMMary]:CONDition?.....117
- :STATus:OPERation:CALibrating[:SUMMary]:ENABle.....118
- :STATus:OPERation:CALibrating[:SUMMary][:EVENt]?.....118
- :STATus:OPERation:CALibrating[:SUMMary]:NTRansition.....118
- :STATus:OPERation:CALibrating[:SUMMary]:PTRansition.....119
- :STATus:OPERation:CONDition?.....119
- :STATus:OPERation:ENABle.....120
- :STATus:OPERation[:EVENt]?.....120
- :STATus:OPERation:NTRansition.....121
- :STATus:OPERation:PTRansition.....121
- :STATus:OPERation:LLFail[:SUMMary]:CONDition?.....121
- :STATus:OPERation:LLFail[:SUMMary]:ENABle.....122
- :STATus:OPERation:LLFail[:SUMMary][:EVENt]?.....122

- :STATus:OPERation:LLFail[:SUMMary]:NTRansition 122
- :STATus:OPERation:LLFail[:SUMMary]:PTRansition 123
- :STATus:OPERation:SENSe[:SUMMary]:CONDition? 123
- :STATus:OPERation:SENSe[:SUMMary]:ENABle 124
- :STATus:OPERation:SENSe[:SUMMary][:EVENt]? 124
- :STATus:OPERation:SENSe[:SUMMary]:NTRansition 124
- :STATus:OPERation:SENSe[:SUMMary]:PTRansition 125
- :STATus:OPERation:TRIGger[:SUMMary]:CONDition? 126
- :STATus:OPERation:TRIGger[:SUMMary]:ENABle 126
- :STATus:OPERation:TRIGger[:SUMMary][:EVENt]? 126
- :STATus:OPERation:TRIGger[:SUMMary]:NTRansition 126
- :STATus:OPERation:TRIGger[:SUMMary]:PTRansition 127
- :STATus:OPERation:ULFail[:SUMMary]:CONDition? 127
- :STATus:OPERation:ULFail[:SUMMary]:ENABle 128
- :STATus:OPERation:ULFail[:SUMMary][:EVENt]? 128
- :STATus:OPERation:ULFail[:SUMMary]:NTRansition 128
- :STATus:OPERation:ULFail[:SUMMary]:PTRansition 129
- :STATus:PRESet 129
- :STATus:QUEStionable:CALibration[:SUMMary]:CONDition? 130
- :STATus:QUEStionable:CALibration[:SUMMary]:ENABle 130
- :STATus:QUEStionable:CALibration[:SUMMary][:EVENt]? 131
- :STATus:QUEStionable:CALibration[:SUMMary]:NTRansition 131
- :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition 132
- :STATus:QUEStionable:CONDition? 132
- :STATus:QUEStionable:ENABle 132
- :STATus:QUEStionable[:EVENt]? 133
- :STATus:QUEStionable:NTRansition 133
- :STATus:QUEStionable:PTRansition 134
- :STATus:QUEStionable:POWer[:SUMMary]:CONDition? 134
- :STATus:QUEStionable:POWer[:SUMMary]:ENABle 134
- :STATus:QUEStionable:POWer[:SUMMary][:EVENt]? 135
- :STATus:QUEStionable:POWer[:SUMMary]:NTRansition 135
- :STATus:QUEStionable:POWer[:SUMMary]:PTRansition 136

Table 3.10 Commands or events affecting the state register

State register	*RST	*CLS	Start up	STATus:PRESet
SCPI transition filter (NTR and PTR)	No effect	No effect	Preset	Preset
SCPI enable register	No effect	No	Preset	Preset
SCPI event register	No effect	Clear	Clear	No effect

SCPI error/event queue enable	No effect	No effect	Preset	Preset
SCPI error/event queue	No effect	Clear	Clear	No effect
IEEE488.2 Register ESE SRE	No effect	No	Clear	No effect
IEEE488.2 Register SESR STB	No effect	Clear	Clear	No effect

Preset state description: The preset value of PTR is 0x7fff (32767); both NTR and enable registers are cleared.

- Command Sets....94
- Transition Filter....95
- Device state register description....95
- Operation state register description....98
- Calibration operation state register description....99
- Lower limit detection operation state register description....101
- Sensing operation state register description....103
- Trigger operation state register description....105
- Upper limit detection operation state register description....107
- Question state register description....109
- Calibration question state register description....111
- Power question state register description....113

3.3.8.1 Command sets

The content of the state register can be queried or set with the following command set:

:CONDition?

Query the value of the condition register of the state register. The return format is < NR1 > . The range is from 0 to 32767. After the query, the value of the condition register remains unchanged.

:ENABle <NRf>|<non-decimal number>

Query or set the event enable register of the state register, and the highest bit (bit15) is always 0.

[:EVENt?]

Query the event register of the state register, and clear the register after the query.

:NTRansition <NRf>|<non-decimal number>

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

:PTRansition <NRf>|<non-decimal number>。

Query or set the positive transition filter of the state register, and the highest bit is always 0.

The state registers supported in this case include:

STATus:DEVice

STATus:OPERation
STATus:OPERation:CALibrating[:SUMMary]
STATus:OPERation:LLFail[:SUMMary]
STATus:OPERation:SENSe[:SUMMary]
STATus:OPERation:TRIGger[:SUMMary]
STATus:OPERation:ULFail[:SUMMary]
STATus:QUEStionable
STATus:QUEStionable
STATus:QUEStionable:CALibration[:SUMMary]
STATus:QUEStionable:POWer[:SUMMary]

For example:

:CONDition?can be used to query the calibration operation registerSTATus: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

3.3.8.2 Transition Filter

Refer to Section 9.2 of SCPI-99 for a description of the transition filter. A brief description is given below.

1) Positive transition (PTR): When the condition is set to TRUE by the FALSE event.
2) Negative transition (NTR): When the condition goes from TRUE to FALSE, the event is set to TRUE.
3) Positive transition or negative transition: The event is set to TRUE when the condition goes from FALSE to TRUE or TRUE to FALSE.
4) Clearing the positive transition and negative transition registers will disable event reporting.

3.3.8.3 Device state register description (STATus:DEVice)

Table 3.11 Device state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A detection state
2	4	Channel B detection state (for dual-channel only)
3	8	Channel A error state
4	16	Channel B error state (for dual-channel only)
5	32	USB channel detection state

		(reserved)
6	64	USB channel error state (reserved)
7-14	-	Unused
15	-	Always 0

1) Bits 1 and 2 indicate the detection state of channel A and channel B, respectively.
   a) Return value of STATus:DEVice:CONDition?: 1 indicates that it is detected, and 0 indicates that it is not detected.
   b) Return value of STATus:DEVice[:EVENt]?: 1 indicates that a connection or removal event has occurred. 0 means not occurred. After the query, the event register will be cleared.
   c) When STATus:DEVice:NTRansition is set to 1, STATus:DEVice[:EVENt] will be set to 1 if shiftis detected.
   d) When STATus:DEVice:PTRansition is set to 1, STATus:DEVice[:EVENt] will be set to 1 if access is detected.
2) Bits 3 and 4 indicate errors for channel A and channel B of the 87237, respectively. 1 means error, and 0 means no error found.

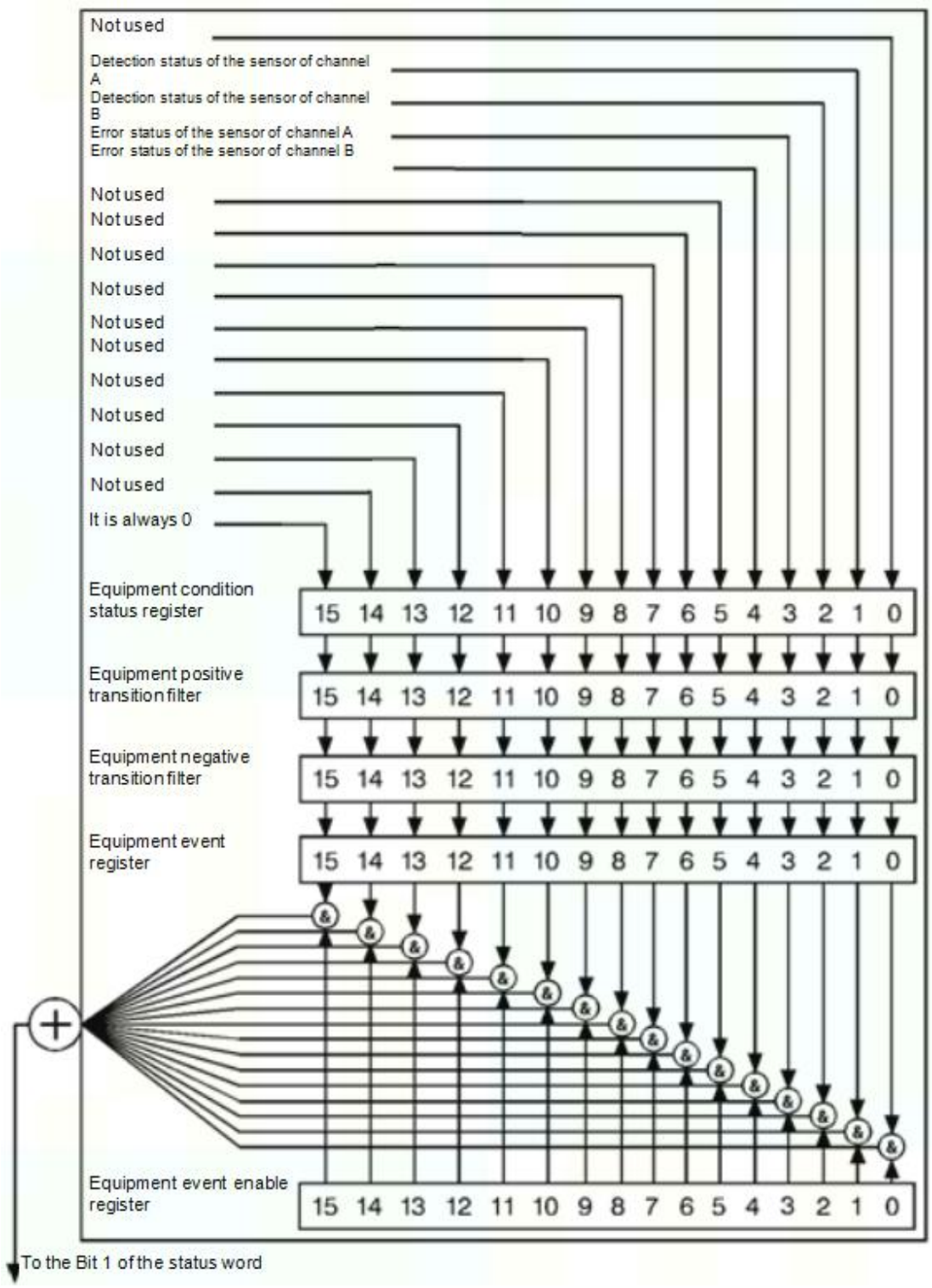


Figure 3.3 Device state register

3.3.8.4 Operation state register description (STATus:OPERation)

Table 3.12 Operation state register description

Bit	Value	Definition
0	1	Calibration summary
1-4	-	Unused
5	32	Waiting for trigger summary
6-9	-	Unused
10	1024	Sensing summary
11	2048	Lower-limit detection failure summary (LLF)
12	4096	Upper-limit detection failure summary (ULF)
13-14	-	Unused
15	-	Always 0

Six groups of operation registers are included:

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

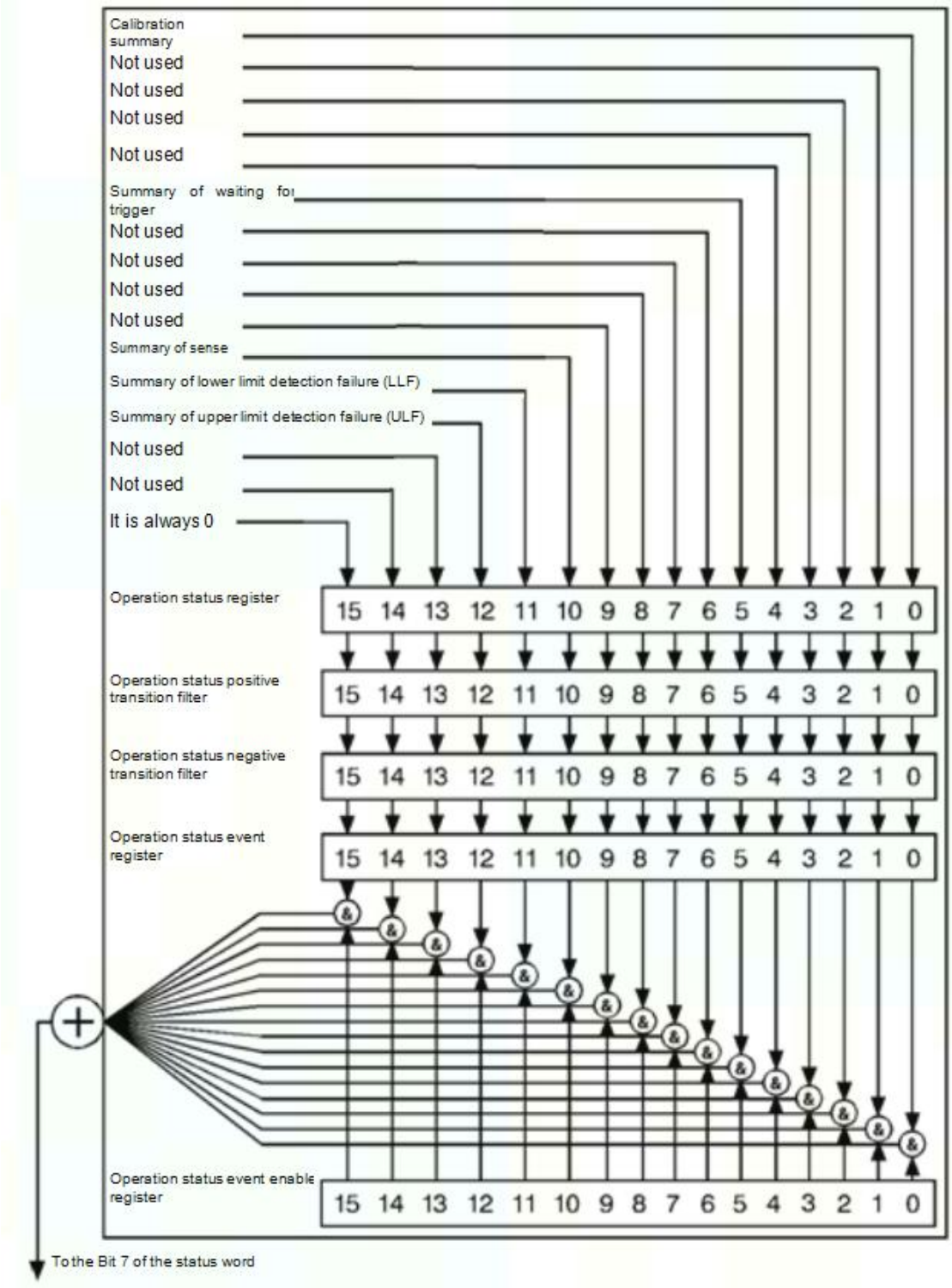


Figure 3.4 Operation state register description

3.3.8.5 Calibration operation state register description (STATus:OPERation:CALibrating[:SUMMary])

Table 3.13 Calibration operation state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A calibration state.
2	4	Channel B calibration state. (for dual-channel only)
3-14	-	Unused
15	-	Always 0

Bit 1 and bit 2 indicate the calibration state of channel A and channel B respectively. 1 indicates that it is being zeroed or calibrated, and 0 indicates that the calibration is completed or not zeroed or calibrated.

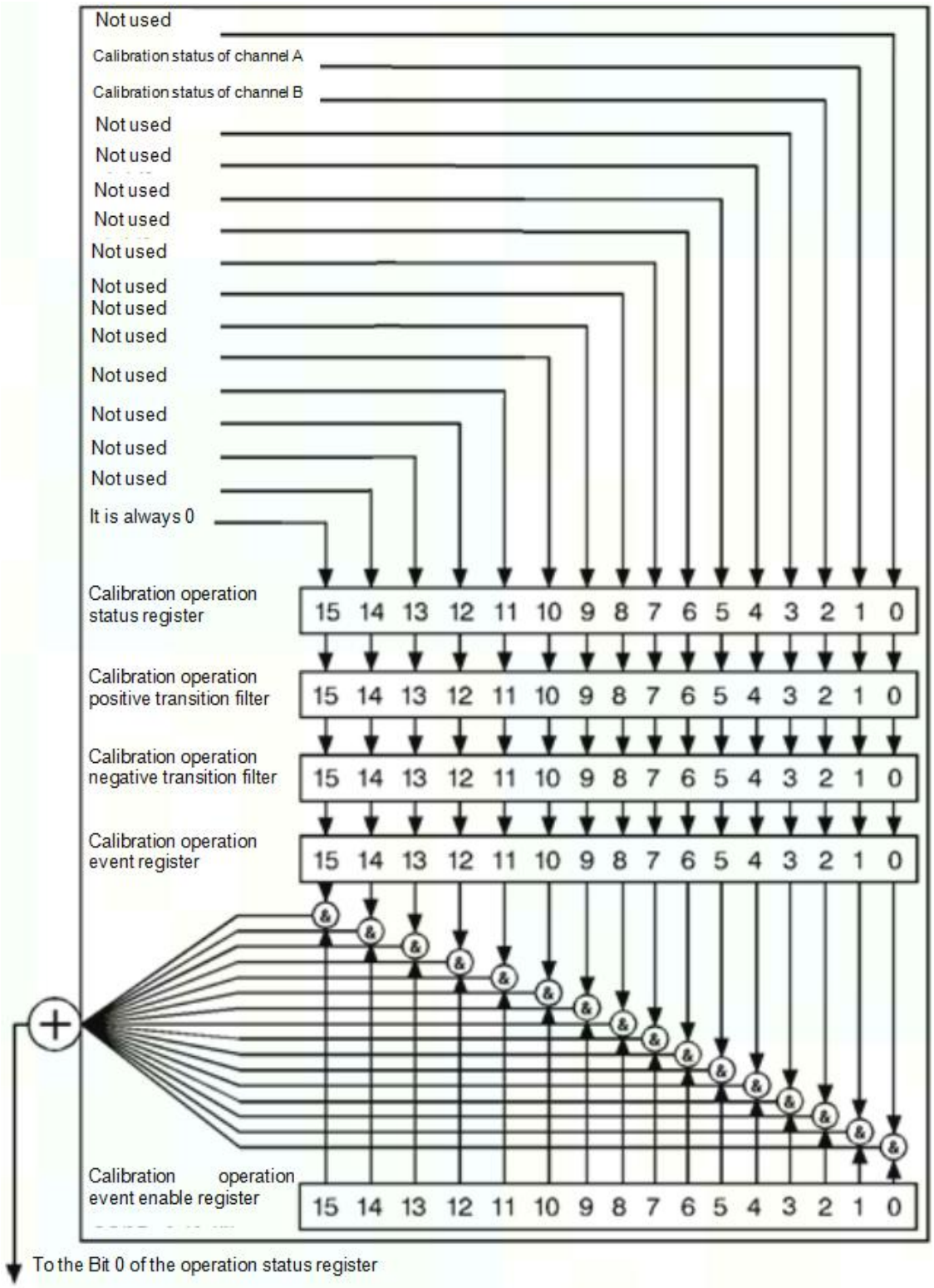


Figure 3.5 Calibration operation state register

3.3.8.6 Lower limit operation state register description (STATus:OPERation:CALibrating[:SUMMary])

Table 3.14 Lower limit detection operation state register description

Bit	Value	Definition
0-2	-	Unused
3	8	Lower limit detection state of measurement 1
4	16	Lower limit detection state of measurement 2
5	32	Lower limit detection state of measurement 3
6	64	Lower limit detection state of measurement 4
7-14	-	Unused
15	-	Always 0

Bit 3 to bit 6 indicates the lower limit failure detection state of the corresponding measurement. 1 indicates that the lower limit is exceeded

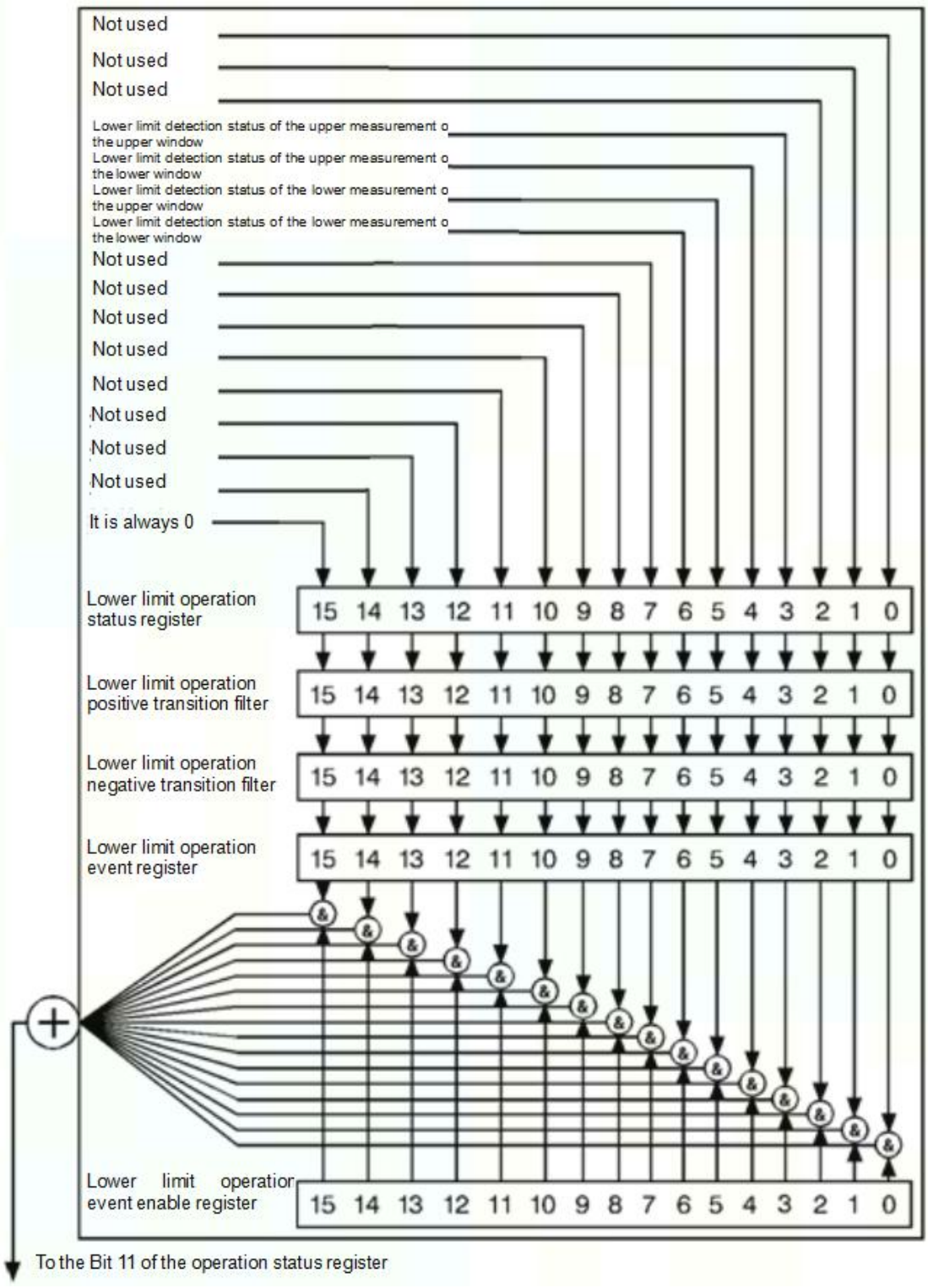


Table 3.6 Lower limit detection operation state register

3.3.8.7 Sensing operation state register description (STATus:OPERation:SENSe[:SUMMary])

Table 3.15 Sensing operation state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A reads EEPROM state
2	4	Channel B reads EEPROM state (for dual-channel only)
3-14	-	Unused
15	-	Always 0

Bits 1 and 2 indicate that channel A and channel B read EEPROM state, respectively, and 1 indicates that the 87237 is being read.

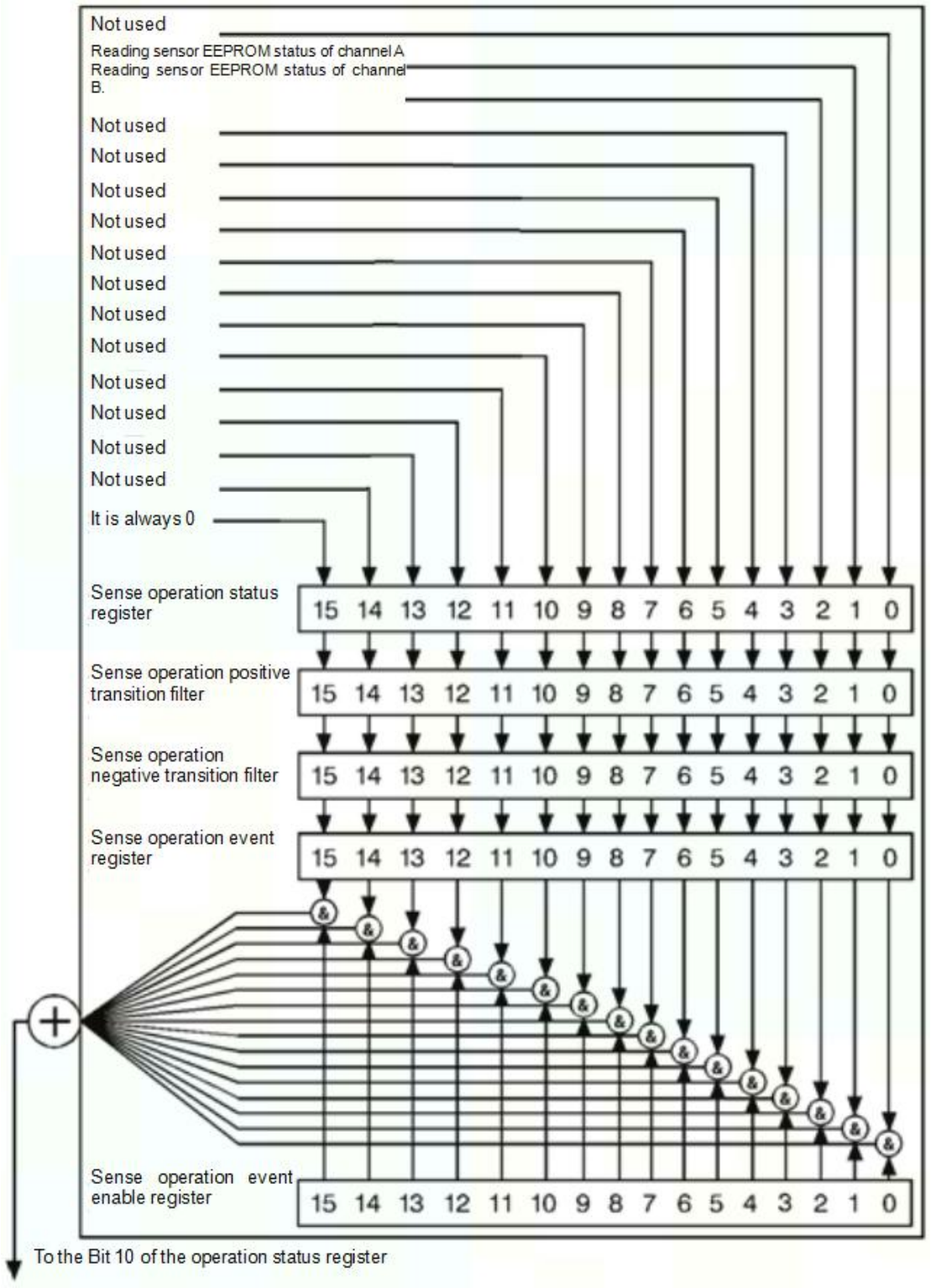


Figure 3.7 Sensing operation state register

3.3.8.8 Trigger operation state register description (STATus:OPERation TRIGger[:SUMMary])

Table 3.16 Trigger operation state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A trigger state
2	4	Channel B trigger state (for dual-channel only)
3-14	-	Unused
15	-	Always 0

Bit 1 and bit 2 indicate the "wait for trigger" state of channel A and channel B respectively, and 1 indicates that it is waiting for trigger.

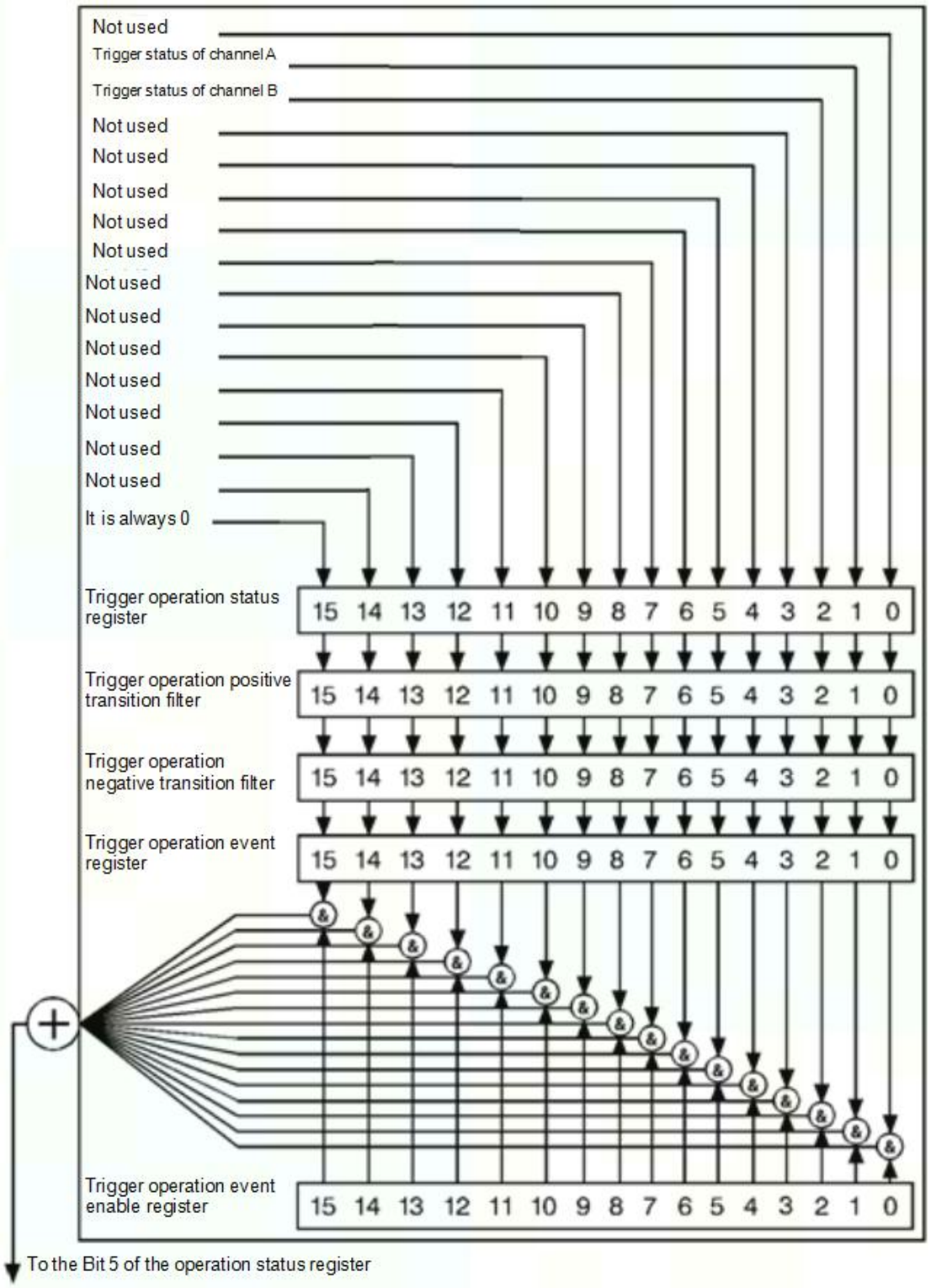


Figure 3.8 Trigger operation state register

3.3.8.9 Upper limit operation state register description (STATus:OPERation:CALibrating[:SUMMary])

Table 3.17 Upper limit detection operation state register description

Bit	Value	Definition
0-2	-	Unused
3	8	Upper limit detection state of measurement 1
4	16	Upper limit detection state of measurement 2
5	32	Upper limit detection state of measurement 3
6	64	Upper limit detection state of measurement 4
7-14	-	Unused
15	-	Always 0

Bit 3 to bit 6 indicates the upper limit failure detection state of the corresponding measurement. 1 indicates that the upper limit is exceeded

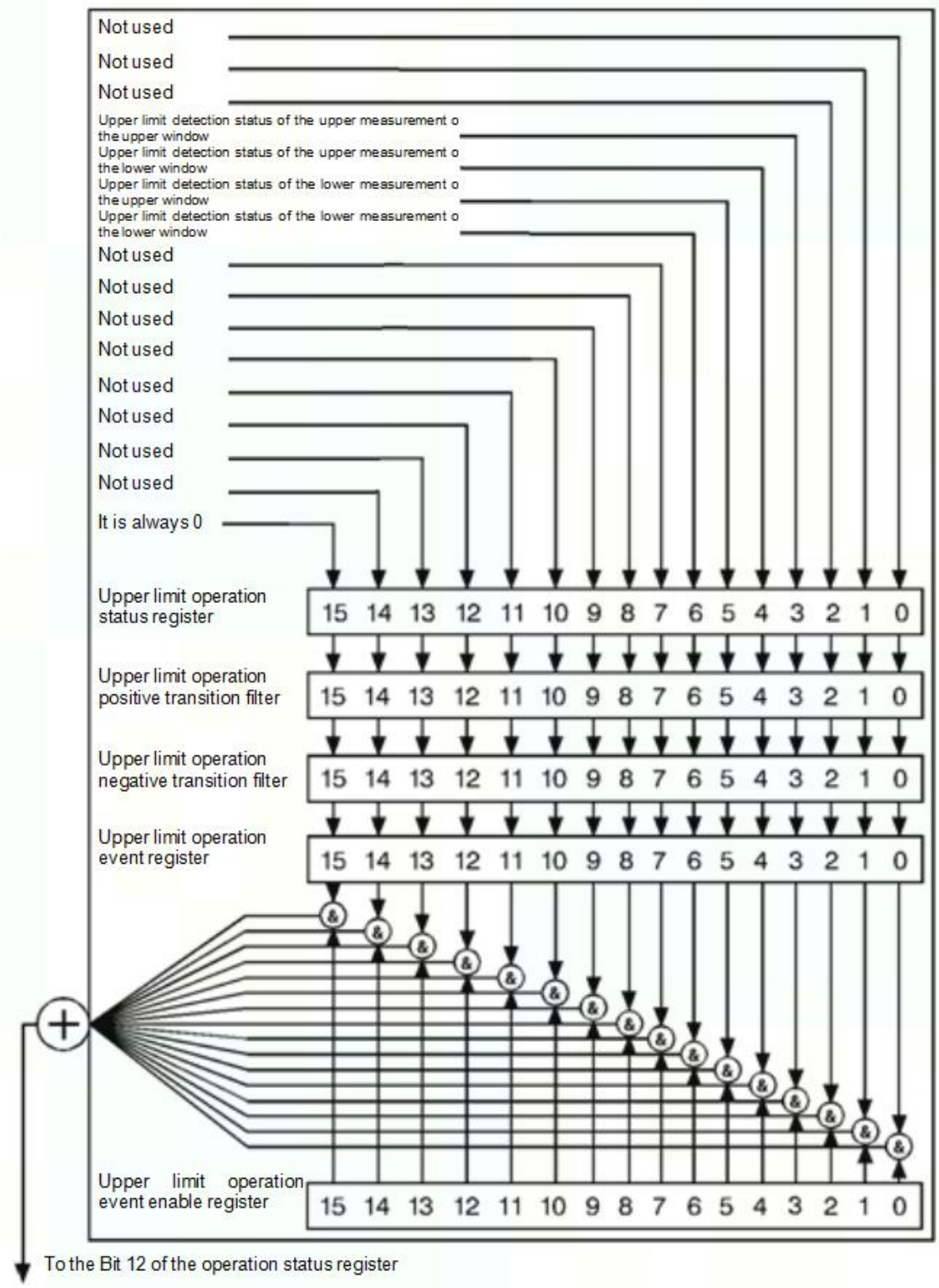


Table 3.9 Upper limit detection operation state register

3.3.8.10 Question state register description (STATus: QUEStionable)

Table 3.18 Question state register description

Bit	Value	Definition
0-2	-	Unused
3	8	Power Summary
4-7	-	Unused
8	256	Calibration Summary
9	512	Power-on self-test
10-14	-	Unused
15	-	Always 0

Three groups of operation registers are included:

STATus:QUEStionable

STATus:QUEStionable:Power[:SUMMary]

STATus:QUEStionable:CALibration[:SUMMary]

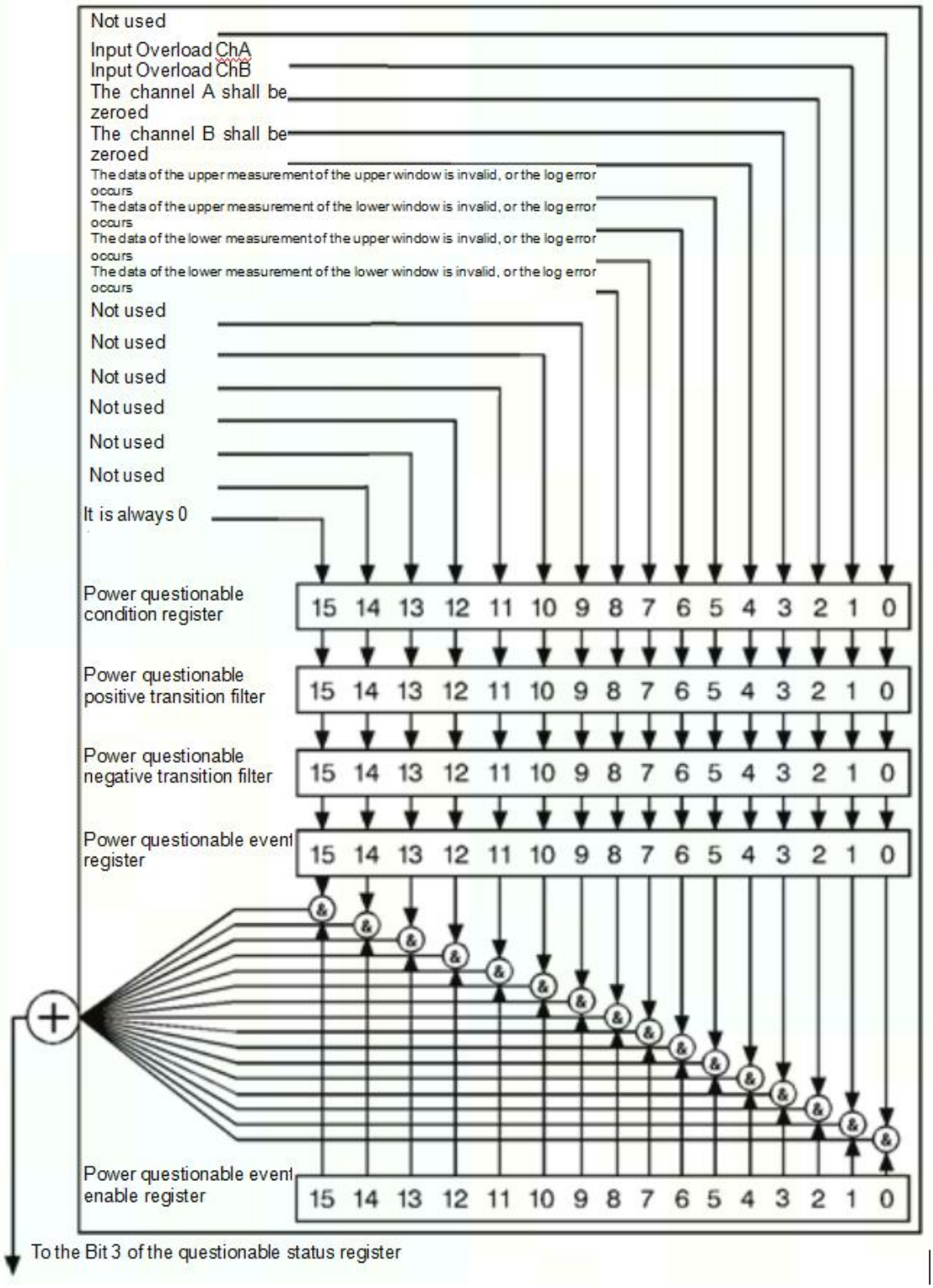


Figure 3.10 Question state register

3.3.8.11 Calibration question operation state register description (STATus QUEStionable:CALibrating[:SUMMary])

Table 3.19 Question state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A zeroing and calibration error
2	4	Channel B zeroing and calibration error (for dual-channel only)
3-14	-	Unused
15	-	Always 0

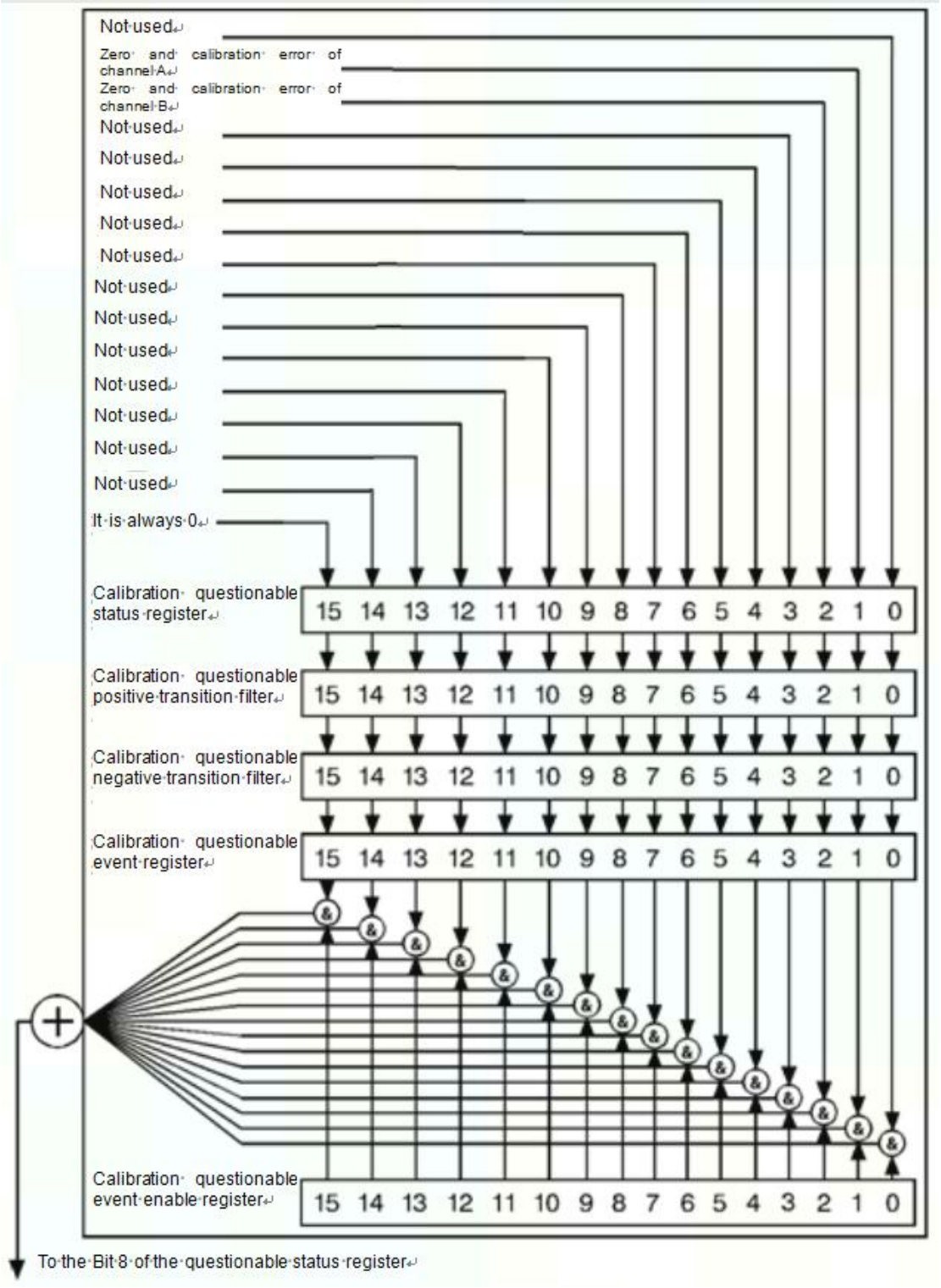


Figure 3.11 Calibration question state register

3.3.8.12 Power question state register description (STATus:QUEStionable:POWer[:SUMMary])

Table 3.20 Power question state register description

Bit	Value	Definition
0	1	Unused
1	2	Channel A input is overloaded
2	4	Channel B input overload (for dual-channel only)
3	8	Channel A needs to be zeroed
4	16	Channel B needs to be zeroed (for dual-channel only)
5	32	Measurement 1 data is invalid, or logarithmically incorrect
6	64	Measurement 2 data is invalid, or logarithmically incorrect
7	128	Measurement 3 data is invalid, or logarithmically incorrect
8	256	Measurement 4 data is invalid, or logarithmically incorrect
9-14	-	Unused
15	-	Always 0

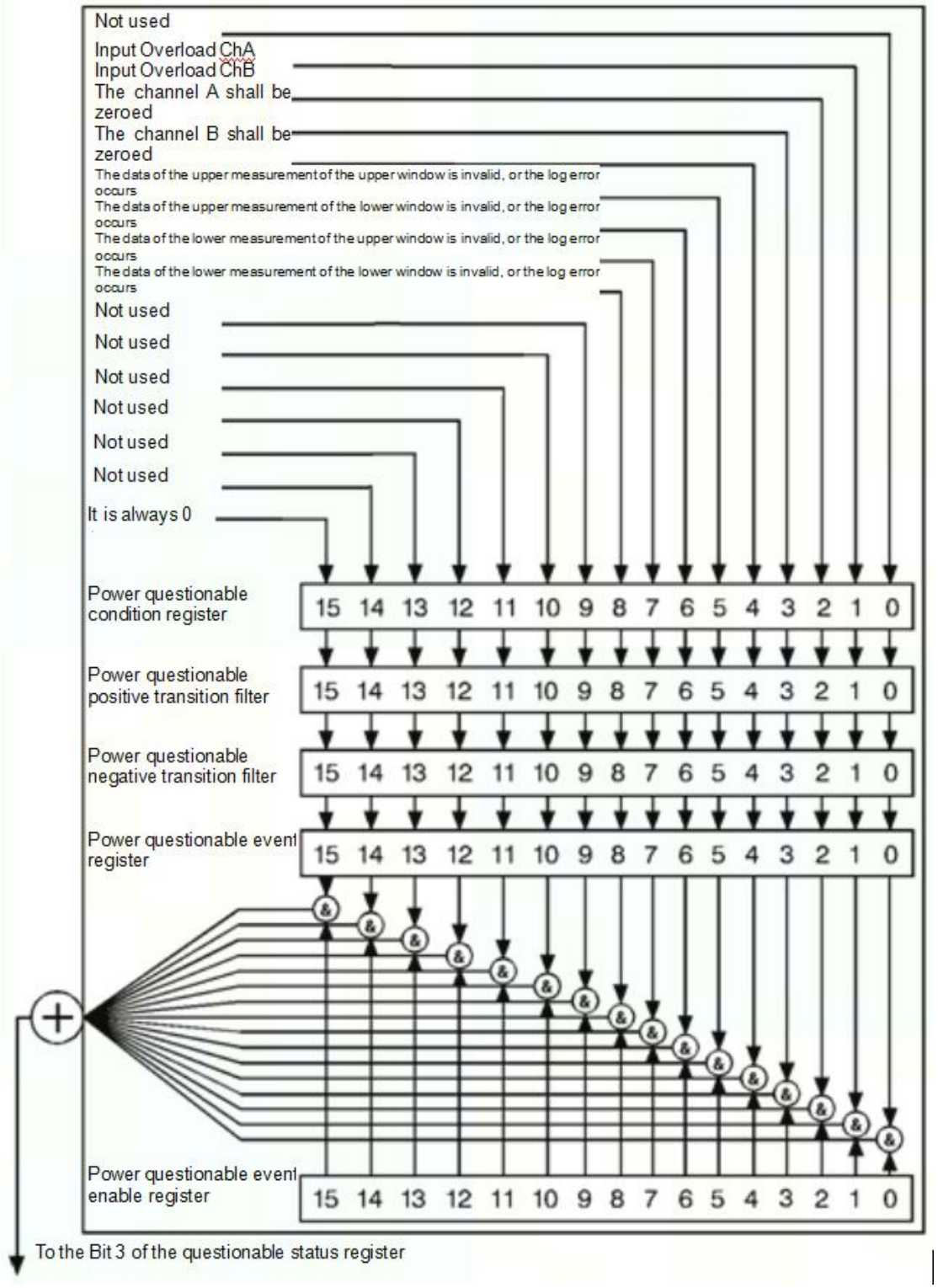


Figure 3.12 Power question state register

:STATus:DEVice:CONDition?

Function: Query the value in the device state condition register.
Query: :STATus:DEVice:CONDition?
Setting: Not supported

Example: STAT:DEV:COND?
Limit: No
Error message: No
Reset state: No

:STATus:DEVice:ENABle

Function: Query or set the device state event enable register. Operate by bit. 0 means that reporting state events to bit 1 of the higher level state word is disabled. 1 indicates enable.
Query: :STATus:DEVice:ENABle?
Setting: :STATus:DEVice:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example:

STAT:DEV:ENAB?	Query the device state event enable register.
STAT:DEV:ENAB 6	Enable bit 1 and bit 2, which allows reporting of the 87237 detection events to the state word.
STAT:DEV:ENAB #B0110	The meaning is the same as above, using binary numbers.
STAT:DEV:ENAB #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

:STATus:DEVice[:EVENt]?

Function: Query the device event register. After the query, 87237 automatically clears the register.
Query: :STATus:DEVice[:EVENt]?
Setting: Not supported
Example: STAT:DEV?
Limit: No
Error message: No
Reset state: No

:STATus:DEVice:NTRansition

Function: Query or set the negative transition filter of the device.

Query: :STATus:DEVice:NTRansition?

Setting: :STATus:DEVice:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:DEV:NTR?

STAT:DEV:NTR #H06	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:DEV:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:DEV:NTR 6	The meaning is the same as above, using decimal numbers.

Limit: No

Error message: No

Reset state: No

:STATus:DEVice:PTRansition

Function: Query or set the positive transition filter of the device.

Query: :STATus:DEVice:PTRansition?

Setting: :STATus:DEVice:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:DEV:PTR?

STAT:DEV:PTR #H06	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:DEV:PTR #B0110	The meaning is the same as above, using binary numbers.
STAT:DEV:PTR 6	The meaning is the same as above, using decimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the value in the calibration operation state condition register.
Among the returned values, if bit 1 is non-zero, channel A is being zeroed or calibrated; if bit 2 is non-zero, channel B is being zeroed or calibrated. For example returning 2 indicates that channel A is being zeroed or calibrated.

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

Setting: Not supported

Example: STAT:OPER:CAL:COND?

Limit: No
Error message: No
Reset state: No

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

Function: Query or set the calibration operation event enable register. Operate by bit. 0 means that reporting of state events to bit 0 of the operation state is disabled. 1 indicates enable.
Query: :STATus:OPERation:CALibrating[:SUMMary]:ENABle?
Setting: :STATus:OPERation:CALibrating[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:CAL:ENAB?

STAT:OPER:CAL:ENAB	6	Enable bit 1 and bit 2, which allows reporting of calibration operation events to the operation state.
STAT:OPER:CAL:ENAB	#B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:CAL:ENAB	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the calibration operation event register. After the query, 87237 automatically clears the register.
Query: :STATus:OPERation:CALibrating[:SUMMary][:EVENt]?
Setting: Not supported
Example: STAT:OPER:CAL?
Limit: No
Error message: No
Reset state: No

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

Function: Query or set the negative transition filter for calibration operation.
Query: :STATus:OPERation:CALibrating[:SUMMary]:NTRansition?

Setting: :STATus:OPERation:CALibrating[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:CAL:NTR?

STAT:OPER:CAL:NTR 6	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:CAL:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:CAL:NTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the positive transition filter for calibration operation.

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

Setting: :STATus:OPERation:CALibrating[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:CAL:PTR?

STAT:OPER:CAL:PTR 6	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:CAL:PTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:CAL:PTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

:STATus:OPERation:CONDition?

Function: Query the value in the operation state condition register.
Among the returned values, if bit 0 is non-zero, a calibration event is detected; if bit 5 is non-zero, a wait-for-trigger event is detected; if bit 10 is non-zero, an access to or removal event from 87237 is detected; if bit 11 is non-zero, a lower limit detection event is detected; if bit 12 is non-zero, an upper limit detection event is detected.
For example, if a calibration event is detected by this set of registers, the

corresponding bits of the calibration operation enable registers (STAT:OPER:CAL:ENAB) need to be set to non-zero. Others are the same.

Query: :STATus:OPERation:CONDition?

Setting: Not supported

Example: STAT:OPER:COND?

Limit: No

Error message: No

Reset state: No

:STATus:OPERation:ENABle

Function: Query or set the operation state event enable register. Operate by bit. 0 means that reporting state events to bit 7 of the state word is disabled. 1 indicates enable.

Query: :STATus:OPERation:ENABle?

Setting: :STATus:OPERation:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:ENAB?

STAT:OPER:ENAB 1	Enable bit 1, which allows reporting of operation events into the state word.
STAT:OPER:ENAB #B0001	The meaning is the same as above, using binary numbers.
STAT:OPER:ENAB #H01	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

:STATus:OPERation[:EVENt]?

Function: Query the operation state event register. After the query, 87237 automatically clears the register.

Query: :STATus:OPERation[:EVENt]?

Setting: Not supported

Example: STAT:OPER?

Limit: No

Error message: No

Reset state: No

:STATus:OPERation:NTRansition

Function: Query or set the negative transition filter for operation state.
Query: :STATus:OPERation:NTRansition?
Setting: :STATus:OPERation:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:NTR?

STAT:OPER:NTR 1	Negative transition filters the state of bit 0 into the event register.
STAT:OPER:NTR #B0001	The meaning is the same as above, using binary numbers.
STAT:OPER:NTR #H01	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

:STATus:OPERation:PTRansition

Function: Query or set the positive transition filter for operation state.
Query: :STATus:OPERation:NTRansition?
Setting: :STATus:OPERation:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:NTR?

STAT:OPER:NTR #H06	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:NTR 6	The meaning is the same as above, using decimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the value in the lower limit detection operation state condition register.
Among the return values, if bit 3 is non-zero, it indicates the lower limit detection operation failure state of measurement 1; if bit 4 is non-zero, it indicates the lower limit detection operation failure state of measurement 2; if bit 5 is non-zero, it

indicates the lower limit detection operation failure state of measurement 3; if bit 6 is non-zero, it indicates the lower limit detection operation failure state of measurement 4.

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

Setting: Not supported

Example: STAT:OPER:LLF:COND?

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the event enable register for lower limit detection operation. Operate by bit. 0 means that reporting of state events to bit 11 of the operation state is disabled. 1 indicates enable.

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

Setting: :STATus:OPERation:LLFail[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:LLF:ENAB?

STAT:OPER:LLF:ENAB 8	Enable bit 3, which allows reporting of lower limit operation events to the operation state.
STAT:OPER:LLF:ENAB #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:LLF:ENAB #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the event register for lower limit detection operation. After the query, 87237 automatically clears the register.

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

Setting: Not supported

Example: STAT:OPER:LLF?

Limit: No

Error message: No

Reset No

state:

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

Function: Query or set the negative transition filter for lower limit detection operation.
Query: :STATus:OPERation:LLFail[:SUMMary]:NTRansition?
Setting: :STATus:OPERation:LLFail[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:LLF:NTR?

STAT:OPER:LLF:NTR 8	Negative transition filters the state of bit 3 into the event register.
STAT:OPER:LLF:NTR #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:LLF:NTR #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query or set the positive transition filter for lower limit detection operation.
Query: :STATus:OPERation:LLFail[:SUMMary]:PTRansition?
Setting: :STATus:OPERation:LLFail[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:LLF:PTR?

STAT:OPER:LLF:PTR 8	Positive transition filters the state of bit 3 into the event register.
STAT:OPER:LLF:PTR #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:LLF:PTR #H08	The meaning is the same as above, using decimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the value in the sensor operation state condition register.
Among the returned values, if bit 1 is non-zero, channel A reads EEPROM state; If bit 2 is non-zero, channel B reads EEPROM state.

Query: :STATus:OPERation:SENSe[:SUMMary]:CONDition?
Setting: Not supported
Example: STAT:OPER:SENS:COND?
Limit: No
Error message: No
Reset state: No

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

Function: Query or set the sensor operation event enable register. Operate by bit. 0 indicates that the sensing operation state event is disabled into bit 10 of the operation state. 1 indicates enable.
Query: :STATus:OPERation:SENSe[:SUMMary]:ENABle?
Setting: :STATus:OPERation:SENSe[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:SENS:ENAB?

STAT:OPER:SENS:ENAB	6	Enable bit 1 and bit 2, which allows reporting of sensing operation events to the operation state.
STAT:OPER:SENS:ENAB	#B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:SENS:ENAB	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the sensing operation event register. After the query, 87237 automatically clears the register.
Query: :STATus:OPERation:SENSe[:SUMMary][:EVENt]?
Setting: Not supported
Example: STAT:OPER:SENS?
Limit: No
Error message: No
Reset state: No

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

Function: Query or set the negative transition filter for sensing operation.

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

Setting: :STATus:OPERation:SENSe[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:SENS:NTR?

STAT:OPER:SENS:NTR 6	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:SENS:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:SENS:NTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the positive transition filter for sensing operation.

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

Setting: :STATus:OPERation:SENSe[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:SENS:PTR?

STAT:OPER:SENS:PTR 6	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:SENS:PTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:SENS:PTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the value in the trigger operation state condition register.
Among the returned values, if bit 1 is non-zero, channel A is waiting for trigger; If bit 2 is non-zero, channel B is waiting for trigger. For example, returning 2 indicates that channel A is waiting for trigger.

Query: :STATus:OPERation:CALibrating[:SUMMary]:CONDition?
Setting: Not supported
Example: STAT:OPER:CAL:COND?
Limit: No
Error message: No
Reset state: No

:STATus:OPERation:TRIGger[:SUMMary]:ENABle

Function: Query or set the trigger operation event enable register. Operate by bit. 0 means that reporting of state events to bit 5 of the operation state is disabled. 1 indicates enable.
Query: :STATus:OPERation:TRIGger[:SUMMary]:ENABle?
Setting: :STATus:OPERation:TRIGger[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:TRIG:ENAB?

STAT:OPER:TRIG:ENAB	6	Enable bit 1 and bit 2, which allows reporting of trigger operation events to the operation state.
STAT:OPER:TRIG:ENAB	#B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:TRIG:ENAB	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the trigger operation event register. After the query, 87237 automatically clears the register.
Query: :STATus:OPERation:TRIGger[:SUMMary][:EVENt]?
Setting: Not supported
Example: STAT:OPER:TRIG?
Limit: No
Error message: No
Reset state: No

:STATus:OPERation:TRIGger[:SUMMary]:NTRansition

Function: Query or set the negative transition filter for trigger operation.

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

Setting: :STATus:OPERation:TRIGger[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:TRIG:NTR?

STAT:OPER:TRIG:NTR 6	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:TRIG:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:TRIG:NTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

:STATus:OPERation:TRIGger[:SUMMary]:PTRansition

Function: Query or set the positive transition filter for trigger operation.

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

Setting: :STATus:OPERation:TRIGger[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:OPER:TRIG:PTR?

STAT:OPER:TRIG:PTR 6	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:TRIG:PTR #B0110	The meaning is the same as above, using binary numbers.
STAT:OPER:TRIG:PTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the value in the upper limit detection operation state condition register.
Among the returned values, if bit 3 is non-zero, it indicates the upper limit detectior operation failure state of measurement 1; if bit 4 is non-zero, it indicates the upper limit detection operation failure state of measurement 2; if bit 5 is non-zero, it

indicates the upper limit detection operation failure state of measurement 3; if bit 6 is non-zero, it indicates the upper limit detection operation failure state of measurement 4.

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

Setting: Not supported

Example: STAT:OPER:ULF:COND?

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the event enable register for upper limit detection operation. Operate by bit. 0 means that reporting of state events to bit 12 of the operation state is disabled. 1 indicates enable.

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

Setting: :STATus:OPERation:ULFail[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example:

STAT:OPER:ULF:ENAB?	Query the value of the upper limit detection operation event enable register.
STAT:OPER:ULF:ENAB 8	Enable bit 3, which allows reporting of upper limit operation events to the operation state.
STAT:OPER:ULF:ENAB #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:ULF:ENAB #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the event register for upper limit detection operation. After the query, 87237 automatically clears the register.

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

Setting: Not supported

Example: STAT:OPER:ULF?

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the negative transition filter for upper limit detection operation.
Query: :STATus:OPERation:ULFail[:SUMMary]:NTRansition?
Setting: :STATus:OPERation:ULFail[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:ULF:NTR?

STAT:OPER:ULF:NTR 8	Negative transition filters the state of bit 3 into the event register.
STAT:OPER:ULF:NTR #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:ULF:NTR #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query or set the positive transition filter for upper limit detection operation.
Query: :STATus:OPERation:ULFail[:SUMMary]:PTRansition?
Setting: :STATus:OPERation:ULFail[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:OPER:ULF:PTR?

STAT:OPER:ULF:PTR 8	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:OPER:ULF:PTR #B1000	The meaning is the same as above, using binary numbers.
STAT:OPER:ULF:PTR #H08	The meaning is the same as above, using decimal numbers.

Limit: No
Error message: No
Reset state: No

:STATus:PRESet

Function: Preset some state registers as follows, and other registers remain unchanged.
Query: Not supported

Setting: :STATus:PRESet
Example: STAT:PRES
Limit: No
Error message: No
Reset state: No

Table 3.21 Operation state register description

Register	Sub register	Preset state
OPERation	ENABle	All 0
	PTR	All 1
	NTR	All 0
QUEStionable	ENABle	All 0
	PTR	All 1
	NTR	All 0
Other	ENABle	All 0
	PTR	All 1
	NTR	All 0

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

Function: Query the value in the calibration question state condition register.
Among the returned values, if bit 1 is non-zero, channel A is zeroed and calibratec incorrectly; if bit 2 is non-zero, channel B is zeroed or calibrated incorrectly. For example, returning 2 indicates that channel A is zeroed or calibrated incorrectly.
Query: :STATus:OPERation:CALibrating[:SUMMary]:CONDition?
Setting: Not supported
Example: STAT:OPER:CAL:COND?
Limit: No
Error message: No
Reset state: No

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

Function: Query or set the calibration question event enable register. Operate by bit. C indicates that reporting state events to bit 8 of the question state is disabled. 1 indicates enable.
Query: :STATus:QUEStionable:CALibration[:SUMMary]:ENABle?

Setting: :STATus:QUEStionable:CALibration[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:QUES:CAL:ENAB?

STAT:QUES:CAL:ENAB 6	Enable bit 1 and bit 2, which allows reporting of calibration question events to the question state.
STAT:QUES:CAL:ENAB #B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:CAL:ENAB #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the calibration question event register. After the query, 87237 automatically clears the register.

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

Setting: Not supported

Example: STAT:QUES:CAL?

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the negative transition filter for calibration question.

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

Setting: :STATus:QUEStionable:CALibration[:SUMMary]:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:QUES:CAL:NTR?

STAT:QUES:CAL:NTR 6	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:QUES:CAL:NTR #B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:CAL:NTR #H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query or set the positive transition filter for calibration question.
Query: :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition?
Setting: :STATus:QUEStionable:CALibration[:SUMMary]:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:QUES:CAL:PTR?

STAT:QUES:CAL:PTR	6	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:QUES:CAL:PTR	#B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:CAL:PTR	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

:STATus:QUEStionable:CONDition?

Function: Query the value in the question state condition register.
Among the returned values, if bit 3 is non-zero, a power question event is detected; if bit 8 is non-zero, a calibration question event is detected; if bit 9 is non-zero, the power-on self-test failed.
For example, if a calibration question event is detected by this set of registers, the corresponding bit of the calibration question enable register (STAT:QUES:CAL:ENAB) needs to be set non-zero. Others are the same.
Query: :STATus:QUEStionable:CONDition?
Setting: Not supported
Example: STAT:QUES:COND?
Limit: No
Error message: No
Reset state: No

:STATus:QUEStionable:ENABle

Function: Query or set the question state event enable register. Operate by bit. 0 means that reporting state events to bit 3 of the state word is disabled. 1 indicates enable.

Query: :STATus:QUEStionable:ENABle?

Setting: :STATus:QUEStionable:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:QUES:ENAB?

STAT:QUES:ENAB 8	Enable bit 1, which allows reporting of calibration question events into the state word.
STAT:QUES:ENAB #B1000	The meaning is the same as above, using binary numbers.
STAT:QUES:ENAB #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

:STATus:QUEStionable[:EVENt]?

Function: Query the question state event register. After the query, 87237 automatically clears the register.

Query: :STATus:QUEStionable[:EVENt]?

Setting: Not supported

Example: STAT:QUES?

Limit: No

Error message: No

Reset state: No

:STATus:QUEStionable:NTRansition

Function: Query or set the negative transition filter for question state.

Query: :STATus:QUEStionable:NTRansition?

Setting: :STATus:QUEStionable:NTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:QUES:NTR?

STAT:QUES:NTR 8	Negative transition filters the state of bit 3 into the event register.
STAT:QUES:NTR #B1000	The meaning is the same as above, using binary numbers.

	STAT:QUES:NTR #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

:STATus:QUEStionable:PTRansition

Function: Query or set the positive transition filter for question state.
Query: :STATus:QUEStionable:PTRansition?
Setting: :STATus:QUEStionable:PTRansition <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.
Example: STAT:QUES:PTR?

	STAT:QUES:PTR 8	Positive transition filters the state of bit 3 into the event register.
	STAT:QUES:PTR #B0110	The meaning is the same as above, using binary numbers.
	STAT:QUES:PTR #H08	The meaning is the same as above, using hexadecimal numbers.

Limit: No
Error message: No
Reset state: No

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

Function: Query the value in the power question state condition register.
Among the returned values,
If bit 1 is non-zero, the channel A input is overloaded;
If bit 2 is non-zero, the channel B input is overloaded;
If bit 3 is non-zero, the channel A needs to be zeroed;
If bit 4 is non-zero, the channel B needs to be zeroed;
If bit 5 is non-zero, the measurement 1 data is invalid, or logarithmically incorrect;
If bit 6 is non-zero, the measurement 2 data is invalid, or logarithmically incorrect;
If bit 7 is non-zero, the measurement 3 data is invalid, or logarithmically incorrect;
If bit 8 is non-zero, the measurement 4 data is invalid, or logarithmically incorrect.
If, for example, 8 is returned, channel A needs to be zeroed.
Query: :STATus:QUEStionable:POWer[:SUMMary]:CONDition?
Setting: Not supported
Example: STAT:QUES:POW:COND?
Limit: No

Error message: No

Reset state: No

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

Function: Query or set the power question event enable register. Operate by bit. 0 indicates that reporting state events to bit 3 of the question state is disabled. 1 indicates enable.

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

Setting: :STATus:QUEStionable:POWer[:SUMMary]:ENABle <NRf>|<non-decimal number>
The parameter range is from 0 to 32767.

Example: STAT:QUES:POW:ENAB?

STAT:QUES:POW:ENAB	6	Enable bit 1 and bit 2, which allows reporting of power question events to the question state.
STAT:QUES:POW:ENAB	#B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:POW:ENAB	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query the power question event register. After the query, 87237 automatically clears the register.

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

Setting: Not supported

Example: STAT:QUES:POW?

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the negative transition filter for power question.

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

Setting: :STATus:QUEStionable:POWer[:SUMMary]:NTRansition <NRf>|< non decimal number >

The parameter range is from 0 to 32767.

Example: STAT:QUES:POW:NTR?

STAT:QUES:POW:NTR	6	Negative transition filters the state of bit 1 and bit 2 into the event register.
STAT:QUES:POW:NTR	#B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:POW:NTR	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

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

Function: Query or set the positive transition filter for power question.

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

Setting: :STATus:QUEStionable:POWer[:SUMMary]:PTRansition <NRf>|< non decimal number >

The parameter range is from 0 to 32767.

Example: STAT:QUES:POW:PTR?

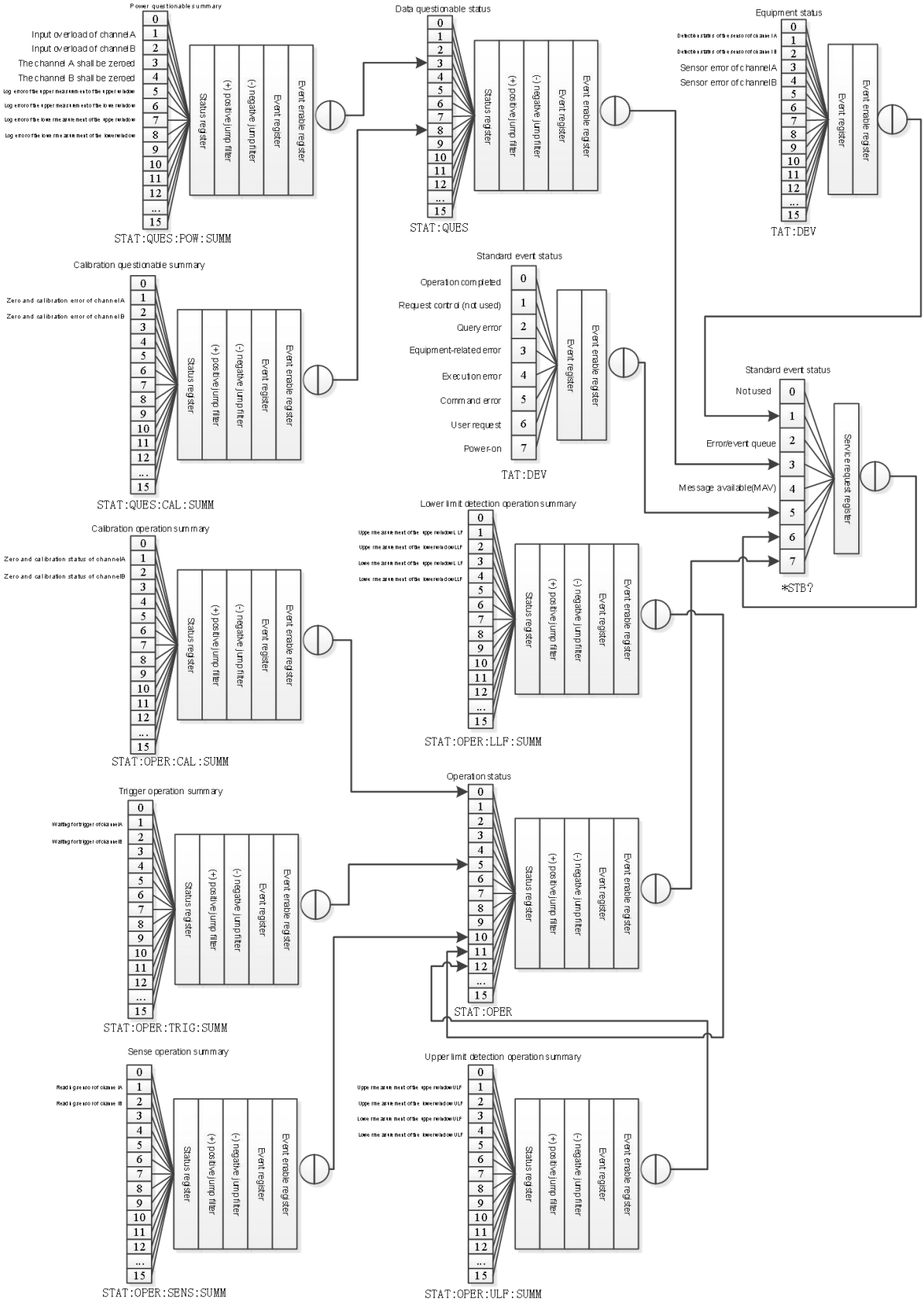
STAT:QUES:POW:PTR	6	Positive transition filters the state of bit 1 and bit 2 into the event register.
STAT:QUES:POW:PTR	#B0110	The meaning is the same as above, using binary numbers.
STAT:QUES:POW:PTR	#H06	The meaning is the same as above, using hexadecimal numbers.

Limit: No

Error message: No

Reset state: No

Attachment: state block diagram


Power questionable summary
Input overload of channel A
Input overload of channel B
The channel A shall be zeroed
The channel B shall be zeroed
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 error of channel A
Sensor error of channel B
TAT:DEV
Calibration questionable summary
Zero and calibration error of channel A
Zero and calibration error of channel B
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?
Calibration operation summary
Zero and calibration status of channel A
Zero and calibration status of channel B
STAT:OPER:CAL:SUMM
Lower limit detection operation summary
STAT:OPER:LLF:SUMM
Trigger operation summary
Waiting for trigger of channel A
Waiting for trigger of channel B
STAT:OPER:TRIG:SUMM
Operation status
STAT:OPER
Sense operation summary
Reading zero of channel A
Reading zero of channel B
STAT:OPER:SENS:SUMM
Upper limit detection operation summary
STAT:OPER:ULF:SUMM

3.3.9 System subsystem command (SYSTem)

- :SYSTem:ERRor:CODE?.....138
- :SYSTem:ERRor[:NEXT]?.....138
- :SYSTem:HELP:HEADers?.....139
- :SYSTem:IDN.....139
- :SYSTem:IDN:AUTO.....139
- :SYSTem:PRESet.....139
- :SYSTem:VERSion?.....142

:SYSTem:ERRor:CODE?

Function: Return the error code from the error queue of the 87237. When an error is generated, the error code is stored in the error queue. Every time this command is executed, this message will be removed from the error queue. The order of error messages out of the queue is first-in-first-out (FIFO), i.e., the oldest messages go out of the queue first. The error queue can be cleared with the *CLS command. Executing this command when the error queue is empty will return 0. The error queue can hold up to 1000 error messages.

Query: :SYSTem:ERRor:CODE?

Setting: Not supported

Example: SYST:ERR:CODE?

Limit: No

Error message: No

Reset state: No

:SYSTem:ERRor[:NEXT]?

Function: Return the error code and error message from the error queue of the 87237. When an error is generated, the error code and error message are stored in the error queue. Every time this command is executed, this message will be removed from the error queue. The order of error messages out of the queue is first-in-first-out (FIFO), i.e., the oldest messages go out of the queue first. The error queue can be cleared with the *CLS command. Executing this command when the error queue is empty will return "0, "No Error"". The error queue can hold up to 1000 error messages.

Query: :SYSTem:ERRor[:NEXT]?

Setting: Not supported

Example: SYST:ERR?

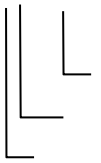
Limit: No

Error message: No

Reset state: No

:SYSTem:HELP:HEADers?

Function: Query the list of commands supported by 87237. The data format is <Arbitrary data block>. See Section 7.7.6 of IEEE 488.2. #nNNN...Nddd……ddd<LF>



Data newline character indicating the end of the data block.
Data length (i.e., the number of bytes of d)
Number of bits of data length (i.e., the number of bits of N)
The marker for the start of the data block.

The data block is of the form #510331 ..... in <LF> n = 5 and N = 10331.

Query: :SYSTem:HELP:HEADers?

Setting: Not supported

Example: SYST:HELP:HEAD?

Limit: No

Error message: No

Reset state: No

:SYSTem:IDN

Function: Query or set the user-defined string of the 87237.

Related programmed control commands ":SYSTem:IDN:AUTO", "*IDN?"

Query: :SYSTem:IDN?

Setting: :SYSTem:IDN <string parameters>
Valid forms of string parameters are "A, B, C, D"
Where A stands for the name of the manufacturer, such as Ceyear
B stands for the product model, such as 87237D
C stands for product serial number, such as SN01
D stands for the version number of the product, such as 1.0.0

Example: SYST:IDN? Return the current user-defined IDN return string.
SYST:IDN "Ceyear,87237D,001,1.0" Set the user-defined IDN return string

Limit: No

Error message: No

Reset state: No

:SYSTem:IDN:AUTO

Function: Enable or disable user-defined strings,
ON: Allows users to query and set user-defined strings via ":SYSTem:IDN", "*IDN?".
OFF: Disable users to query and set user-defined strings via ":SYSTem:IDN" anc "*IDN?".
Related programmed control commands ":SYSTem:IDN", "*IDN?"

Query: :SYSTem:IDN:AUTO?

Setting: :SYSTem:IDN:AUTO <Boolean data>
Valid forms of <Boolean data> are: 0, OFF, 1, ON

Example:

SYST:IDN:AUTO?	Query the ON/OFF state of user-defined string.
SYST:IDN:AUTO 1	Enable user-defined strings.
SYST:IDN:AUTO 0	Disable user-defined strings.

Limit: No

Error message: No

Reset state: OFF

:SYSTem:PRESet

Function: Reset 87237 to the state specified by the parameter. A total of 23 states are offerec in this case.

Query: Not supported

Setting: :SYSTem:PRESet [character data]
The form of [character data] is as follows (DEFault is the default parameter, i.e., the parameter when no parameters are included.) :

DEFault.....140

Example:

SYST:PRES DEF	Reset state to default value.

Limit: No

Error message: No

Reset state: See the table below.

1) DEFault

Table 3.22 Default reset state (DEFault)

Command	Settings	Description
CALC[1]|2|3|4:FEED[1]|2	"POW:AVER"	Set the average measurement mode
CALC[1]|2|3|4:GAIN[:MAGN]	0.000dB	Set display offset to 0

Command	Settings	Description
CALC[1]|2|3|4:GAIN:STAT	OFF	Disable display offset
CALC[1]|2|3|4:LIM:CLE:AUTO	ON	Clear the limit detection state when initializing the measurement
CALC[1]|2|3|4:LIM:LOW[:DATA]	-90dBm	Lower limit
CALC[1]|2|3|4:LIM:STAT	OFF	Disable the measurement limit detection
CALC[1]|2|3|4:LIM:UPP[:DATA]	90dBm	Upper limit
CALC[1]|2|3|4:MATH[:EXPR]	SENS1	Measurement expression
CALC[1]|2|3|4:REL[:MAGN]:AUTO	OFF	Disable the reference value of the relative measurement.
CALC[1]|2|3|4:REL:STAT	OFF	Disable relative measurement.
DISP[:WIND[1]|2][:NUM[1]|2]:RES	3	Measurement display resolution is set to 3
FORM[:READ]:BORD	NORMal	The byte order of binary data is normal.
FORM[:READ][:DATA]	ASCii	The data format is ASCII
INIT[1]|2:CONT	ON	In "wait for trigger" state
MEM:TABL:SEL	No effect	Current frequency response offset table
[SENS[1]]:AVER:COUN:AUTO	ON	Turn on auto average.
[SENS[1]]:AVER:SDET	ON	Turn on the step detection
[SENS[1]]:AVER[:STAT]	ON	Turn on the average
[SENS[1]]:CORR:CSET2[:SEL]	No effect	Currently selected frequency response offset table
[SENS[1]]:CORR:CSET2:STAT	No effect	The ON/OFF state of the currently selected frequency response offset table
[SENS[1]]:CORR:FDOF|GAIN4[:INP][:MAGN]	No effect	Return the frequency response bias value

3.3 Instrument subsystem command

Command	Settings	Description
[SENS[1]]:CORR:GAIN2:STAT	OFF	Disable the channel offset
[SENS[1]]:CORR:GAIN2:[:INPut][:MAGNitude]	0.00dB	Channel offset is 0
[SENS[1]]:FREQ[:CW|:FIX]	50MHz	Channel frequency is 50MHz
TRIG[:SEQ]:DEL	0	Trigger delay
TRIG[:SEQ]:SLOP	POS	Rising edge trigger
TRIG[:SEQ[1]]:SOUR	IMM	Trigger source
UNIT:POW	dBm	Power unit
UNIT:POW:RAT	dB	Rate unit

:SYSTem:VERSion?

Function: Query the SCPI version number used by 87237. The return form is YYYY.X, where YYYY denotes the year and X denotes the version number. It returns 1999.0 in this case.

Query: :SYSTem:VERSion?

Setting: Not supported

Example: SYST:VERS?

Limit: No

Error message: No

Reset state: No

3.3.10 Trigger subsystem command (INITiate/ TRIGger)

- :ABORt[1] 143
- :INITiate[1]:CONTinuous 143
- :INITiate:CONTinuous:ALL 143
- :INITiate:CONTinuous:SEQuence[1] 144
- :INITiate[1][:IMMediate] 144
- :INITiate[:IMMediate]:ALL 145
- :INITiate[:IMMediate]:SEQuence[1] 145
- :INPut:TRIGger:IMPedance 145
- :TRIGger[1][:IMMediate] 145
- :TRIGger[:SEQuence[1]]:COUNt 146
- :TRIGger[:SEQuence[1]]:DELay 146
- :TRIGger[:SEQuence[1]]:IMMediate 147
- :TRIGger[:SEQuence]:SLOPe 147

- :TRIGger[:SEQuence[1]]:SOURce……148

:ABORt[1]

Function: Stop the measurement of the corresponding channel of the 87237. Restart the measurement after sending INITiate:CONTinuous ON.

Query: Not supported

Setting: ABORt[1]

Example:

ABOR	Stop measuring channel A.
ABOR	Stop measuring

Note: 87237 does not set the trigger idle state, i.e. the trigger is always in the wait state.

Limit: No

Error message: No

Reset state: No

:INITiate[1]:CONTinuous

Function: Query or set the trigger state of the 87237: 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.
This command is equivalent to INITiate:CONTinuous:SEQuence[1]

Query: :INITiate[1]:CONTinuous?

Setting: :INITiate[1]:CONTinuous <Boolean data>
OFF | 0 Single
ON|1 Continuous

Example:

INIT:CONT?	Query the trigger state: 0 for single step and 1 for continuous.
INIT:CONT ON	Set channel A to continuous trigger state.

Note: For the parameter OFF, this corresponds to stopping the measurement; for the parameter ON, this corresponds to starting the measurement.

Limit: No

Error message: No

Reset state: Set to ON

:INITiate:CONTinuous:ALL

Function: Query or set the trigger state of all channels of the 87237: 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

ON|1 Continuous

Example: INIT:CONT:ALL? Query the trigger state of all channels. When all channels are continuously triggered, it is 1, otherwise it is 0.

INIT:CONT:ALL ON Set all channels to continuous trigger state.

Limit: No

Error message: No

Reset state: Set to ON

:INITiate:CONTinuous:SEQuence[1]

Function: Query or set the trigger state of the 87237: 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.

This command is equivalent to INITiate[1]:CONTinuous

Query: :INITiate:CONTinuous:SEQuence[1]?

Setting: :INITiate:CONTinuous:SEQuence[1] <Boolean data>

OFF | 0 Single

ON|1 Continuous

Example: INIT:CONT:SEQ? Query the trigger state: 0 for single step and 1 for continuous.

INIT:CONT:SEQ1 ON Set channel A to continuous trigger state.

Limit: No

Error message: No

Reset state: Set to ON

:INITiate[1][:IMMediate]

Function: Set 87237 to "wait for trigger" state. The measurement starts when a trigger event is received.

Equivalent command INITiate:[IMMediate]:SEQuence[1]

Query: Not supported

Setting: :INITiate[1][:IMMediate]

Example: INIT Set 87237 to "wait for trigger" state.

Limit: No
Error message: No
Reset state: No

:INITiate[:IMMediate]:ALL

Function: Set all channels of the 87237 to "wait for trigger" state. The measurement starts when a trigger event is received.
Query: Not supported
Setting: :INITiate[:IMMediate][:ALL]
Example: INIT:ALL — Set all channels of the 87237 to "wait for trigger" state.
Limit: No
Error message: No
Reset state: No

:INITiate[:IMMediate]:SEQuence[1]

Function: Set 87237 to "wait for trigger" state. The measurement starts when a trigger event is received.
Equivalent command:INITiate[1][:IMMediate]
Query: Not supported
Setting: :INITiate[:IMMediate]:SEQuence[1]
Example: INIT:SEQ — Set 87237 to "wait for trigger" state.
Limit: No
Error message: No
Reset state: No

:TRIGger[1][:IMMediate]

Function: This command puts the 87237 in the trigger waiting state immediately.
The equivalent command is :
INITiate[1]:[IMMediate]
TRIGger[:SEQuence[1]][:IMMediate]
Query: Not supported
Setting: :TRIGger[1][:IMMediate]
Example: TRIG — Place the 87237 in the trigger waiting state immediately.

Limit: No

Error message: No

Reset state: No

:TRIGger[:SEQuence[1]]:COUNt

Function: Query or set the number of trigger event detection/measurement cycles.
For example, a trigger count of 100 means that 100 measurements need to be performed in order to respond to 100 triggers.
The "CALibration[1]:ZERO:AUTO" is set to auto-zero OFF when the number of trigger settings exceeds 1.
When "CALibration[1]:ZERO:AUTO" is set to auto-zero ON, the trigger count returns to the default value of 1.

Query: :TRIGger[:SEQuence[1]]:COUNt? [MIN|MAX]

Setting: :TRIGger[:SEQuence[1]]:COUNt <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> ranges from 1 to 500.
DEF is 1,
MIN is 1,
MAX is 80000.

Example:

TRIG:COUN?	Query the trigger times.
TRIG:COUN? MIN	Query the minimum settable trigger times.
TRIG:COUN 100	Set the trigger count to 500.
TRIG:COUN DEF	Set the trigger count to the default value (1 time).

Limit: In the 87237, timeslot list sweep measurement type ([SENSe[1]:]MRATe) shall be set to “FAST”.

Error message: If the current [SENSe[1]:]MRATe is not in FAST, it prompts "-221, "Settings conflict"";

Reset state: trigger times is 1.

:TRIGger[:SEQuence[1]]:DELay

Function: Query or set the trigger delay.

Query: :TRIGger[:SEQuence[1]]:DELay? [MIN|MAX]

Setting: :TRIGger[:SEQuence[1]]:DELay <numeric data>
Valid values for numeric data are: DEF, MIN, MAX, <NRf>, where DEF and <NRf> are used for setting only.
<NRf> ranges from 0 to +1, in seconds,

DEF is 0,
MIN is 0,
MAX is +1.

Example:	TRIG:DEL?	Query the trigger delay.
	TRIG:DEL? MIN	Query the minimum settable trigger delay.
	TRIG:DEL 1E-4	Set the trigger delay to 100us.
	TRIG:DEL DEF	Set the trigger delay to the default value (0 second).

Limit: No
Error message: No
Reset state: Trigger delay is set to 0 second.

:TRIGger[:SEQuence[1]]:IMMediate

Function: This command puts the 87237 in the trigger waiting state immediately.
The equivalent command is:INITiate[1]:[IMMediate]
TRIGger[1][:IMMediate]
Query: Not supported
Setting: :TRIGger[:SEQuence[1]]:IMMediate

Example:	TRIG:IMM	Place channel A in the trigger waiting state immediately.

Limit: No
Error message: No
Reset state: No

:TRIGger[:SEQuence[1]]:SLOPe

Function: Query or set the trigger slope.
Query: :TRIGger[:SEQuence[1]]:SLOPe?
Setting: :TRIGger[:SEQuence[1]]:SLOPe <character data>
Valid values of character data are:
0 or POSitive: Capture the trigger event at the rising edge of the signal.
1 or NEGative: Capture the trigger event at the falling edge of the signal.

Example:	TRIG:SLOP?	Query the trigger slope.
	TRIG:SLOP NEG	Set the falling edge trigger.

Limit: No
Error message: No
Reset The trigger slope is set to rising edge trigger (POSitive).

state:

:TRIGger[:SEQuence[1]]:SOURce

Function: Query or set the trigger source.
The equivalent command is :TRIGger[1]:SOURce
Query: :TRIGger[:SEQuence[1]]:SOURce?
Setting: :TRIGger[:SEQuence[1]]:SOURce <character data>
Valid values of character data are:
BUS: The trigger source is a "*TRG" generic command or a "TRIGger:IMMediate" SCPI command.
EXTernal: The trigger source is the trigger input on the rear panel.
HOLD: Trigger hang, trigger 87237 with "TRIGger:IMMediate" command.
IMMediate: Trigger the system to run all the time. If "INITiate:CONTinous" is set to "ON", the 87237 operates in the free-run mode; otherwise, after "INITiate:IMMediate" is sent, it enters the measurement stop state (IDLE state) after one measurement.
Example:

TRIG:SOUR?	Query the trigger source.
TRIG:SOUR EXT	Set trigger source to External.

Limit: No
Error message: No
Reset state: Trigger source is set to IMM.

3.3.11 Unit subsystem command (UNIT)

- :UNIT[1]|2|3|4:POWer 148
- :UNIT[1]|2|3|4:POWer:RATio 149

:UNIT[1]|2|3|4:POWer

Function: Query or set the specified measurement power unit. The menu operations are linear and logarithmic.
Absolute power measurements are in W and dBm, corresponding to linear and logarithmic, respectively.
Ratio measurements and relative power measurements are in % and dB, corresponding to linear and logarithmic, respectively.
UNIT1 corresponds to measurement 1;
UNIT2 corresponds to measurement 2;
UNIT3 corresponds to measurement 3;
UNIT4 corresponds to measurement 4;
Query: :UNIT[1]|2|3|4:POWer?
Setting: :UNIT[1]|2|3|4:POWer <character data>

Valid character data are:
DBM or 0: Logarithmic display.
W or 1: Linear display

Example:	UNIT2:POW?	Query the power unit of measurement 2.
	UNIT1:POW W	Set the power unit of measurement 1 to W.
Limit:	No	
Error message:	No	
Reset state:	No	

:UNIT[1]|2|3|4:POWer:RATio

Function: Query or set the specified measurement ratio measurement power unit. The menu operations are linear and logarithmic.
UNIT1 corresponds to measurement 1;
UNIT2 corresponds to measurement 2;
UNIT3 corresponds to measurement 3;
UNIT4 corresponds to measurement 4;

Query: :UNIT[1]|2|3|4:POWer:RATio?

Setting: :UNIT[1]|2|3|4:POWer:RATio <character data>
Valid character data are:
DB or 0: Logarithmic display.
PCT or 1: Linear display (PCT indicates %)

Example:	UNIT2:POW:RAT?	Query the power unit of measurement 2.
	UNIT1:POW:RAT PCT	Set the power unit of measurement 1 to PCT.
Limit:	No	
Error message:	No	
Reset state:	No	

3.3.12 Service subsystem command (SERVice)

- :SERVice:BIST:TBASe:STATe 150
- :SERVice:BIST:VIDeo:STATe 150
- :SERVice:SENSor[1]:CDATe? 150
- :SERVice:SENSor[1]:CPLace? 150
- :SERVice:SENSor[1]:FREQuency:MAXimum? 150
- :SERVice:SENSor[1]:FREQuency:MINimum? 150
- :SERVice:SENSor[1]:SNUMber? 151
- :SERVice:SENSor[1]:TYPE? 151

- :SERVice:SNUMber.....151

:SERVice:SENSor[1]:CDATe?

Function: Query the calibration date of the 87237. The calibration date is stored in the EEPROM of the 87237.
Query: :SERVice:SENSor[1]:CDATe?
Setting: Not supported
Example: SERV:SENS:CDAT? Query the calibration date of the 87237.
Limit: No
Error message: No
Reset state: No

:SERVice:SENSor[1]:CPLace?

Function: Query the calibration address of the 87237. The calibration address is stored in the EEPROM of the 87237.
Query: :SERVice:SENSor[1]:CPLace?
Setting: Not supported
Example: SERV:SENS:CPL? Query the calibration address of the 87237.
Limit: No
Error message: No
Reset state: No

:SERVice:SENSor[1]:FREQuency:MAXimum?

Function: Query the maximum frequency of the 87237. The maximum frequency is stored in the EEPROM of the 87237.
Query: :SERVice:SENSor[1]:FREQuency:MAXimum?
Setting: Not supported
Example: SERV:SENS:FREQ:MAX? Query the maximum frequency of the 87237.
Limit: No
Error message: No
Reset state: No

:SERVice:SENSor[1]:FREQuency:MINimum?

Function: Query the minimum frequency of the 87237. The minimum frequency is stored in the

EEPROM of the 87237.

Query: :SERVice:SENSor[1]:FREQuency:MINimum?

Setting: Not supported

Example: SERV:SENS:FREQ:MIN? Query the minimum frequency of the 87237.

Limit: No

Error message: No

Reset state: No

:SERVice:SENSor[1]:SNUMber?

Function: Query the string number of the 87237. The string number of the 87237 is stored ir the EEPROM of the 87237.

Query: :SERVice:SENSor[1]:SNUMber?

Setting: Not supported

Example: SERV:SENS:SNUM? Query the string number of the 87237.

Limit: No

Error message: No

Reset state: No

:SERVice:SENSor[1]:TYPE?

Function: Query the type of the 87237. The type of the 87237 is stored in the EEPROM of the 87237.

Query: :SERVice:SENSor[1]:TYPE?

Setting: Not supported

Example: SERV:SENS:TYPE? Query the type of the 87237.

Limit: No

Error message: No

Reset state: No

4. Programming examples

- Basic Operation Examples.....153
- Advanced Operation Examples.....158
- Application Examples.....163

4.1 Examples of basic operation

Take the following as an example to describe the basic method for realizing instrument programmed control programming via the VISA library. Take the C++ language as an example.

- VISA library.....153
- Examples of runtime environment.....154
- Initializing and setting default state.....155
- Sending setting command.....156
- Reading the state of measuring instrument.....157
- Command synchronization.....157

4.1.1 VISA library

VISA is the generic name for the standard I/O function library and its associated specifications. VISA library function is a set of functions that can be easily called. Its core function can control various types of devices without considering the interface type of devices and the use of different I/O interface software. These library functions are used to write the driver program of the instrument and complete the command and data transmission between the computer and the instrument, so as to realize programmed control of the instrument. By initializing the addressing string ("VISA resource string"), a connection to an instrument with a program port (LAN, USB, GPIB and RS-232, etc.) can be established.

To realize remote control, the VISA library must be installed first. The VISA library is encapsulated with the transmission function at the bottom layer with VXI, GPIB, LAN and USB interfaces, so as to facilitate users' direct calling. The programming interface supported by the 87237 is USB. These interfaces, combined with the VISA library and programming language, allow remote control of the 87237. At present, Keysight I/O Library provided by Keysight is often used as the underlying I/O Library.

Figure 4.1 shows the relationship between the programmed control interfaces, VISA libraries, programming languages and the 87237 with the USB interface as an example.

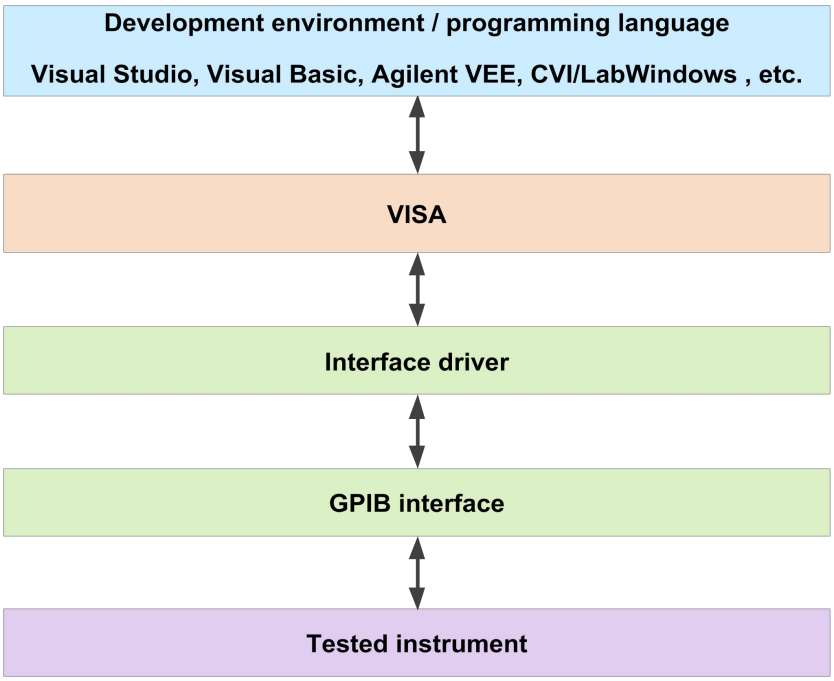


Fig.4.1 Programmable software/hardware layer

4.1.2 Examples of runtime environment

4.1.2.1 Configuration requirements

The programming examples described in this chapter have run successfully on a computer configured as follows.

IBM compatible PC above Pentium class;
Windows 2000, Windows XP, Windows 7 or, Windows 8, Windows 10 operating systems.
Visual Studio 2010/2012/2013/2015/2017 integrated development environment;
VISA library of NI or Keysight;
USB cable

4.1.2.2 Files included

To run an example program written in C/C++, you must include the required files in the VC project.
If you use the VISA library, you must:

Add visa32.lib to the source file;
Add visa.h to the header file.

If you use the NI-488.2 library, you must:

Add GPIB-32.OBJ file to the source file;

Add windows.h file to the header file;

Add Deci-32.h file to the header file.

For more information about the NI-488.2 library and VISA library, please refer to the websites of NI and Keysight respectively.

4.1.3 Initialization and setting default state

To start the program, the VISA resource manager must be initialized, so as to open and establish the communication connection between the VISA library and instrument. The specific steps are as follows:

4.1.3.1 Generation of global variables

Start by generating global variables that other program modules will call, such as instrument handle variables. The following example programs shall contain the following global variables:

```
ViSession iDevHandle;
ViSession iDefaultRM;
const char rgcDevRsc[MAX_RSC_LEN] = "USB0::0x3399::0x3820::2019001::0::INSTR";
const int iTmo = 5000;
```

Where, the constant rgcDevRsc represents the instrument descriptor, "USB0" represents the controller, "0x3399" represents the vendor ID of the instrument, 0x3820 is the product ID, and 2020001 is the product serial number.

If the instrument is controlled via the GPIB interface and the GPIB address is "20", then the value of the variable is

```
const char rgcDevRsc[MAX_RSC_LEN] = "GPIB0::20::INSTR";
```

If the instrument is controlled via a LAN socket interface with IP address 192.168.1.1 and port number 5025, then the value of the variable is

```
const char rgcDevRsc[MAX_RSC_LEN] = "TCPIP0::192.168.1.1::5025::SOCKET";
```

If the instrument is controlled via the LAN interface with IP address 192.168.1.1, then the value of the variable is

```
const char rgcDevRsc[MAX_RSC_LEN] = "TCPIP0::192.168.1.1::INSTR";
```

4.1.3.2 Initialization of the controller

```
/*************************************************************************
```

The following examples show how to open and establish a communication connection between the VISA library and the instrument (as specified by the instrument descriptor).

4.1 Examples of basic operation

//Initialize the master: open the default resource manager and return the instrument handle iDevHandle.

**********/

```
void InitController()
{
    ViStatus iStatus;
    iStatus = viOpenDefaultRM(&iDefaultRM);
    iStatus = viOpen(iDefaultRM, rgcDevRsc, VI_NULL, VI_NULL, &iDevHandle);
}
```

4.1.3.3 Initialize the network analyzer

/**********/

The following example shows how to initialize the default state of the instrument and empty the state register.

/**********/

```
void InitDevice()
{
    ViStatus iStatus;
    ViUInt32 uiRetCnt;
    iStatus = viWrite(iDevHandle, "*CLS\n", strlen("*CLS\n"), &uiRetCnt); //Status Reset
    iStatus = viWrite(iDevHandle, "*RST\n", strlen("*RST\n"), &uiRetCnt); //Instrument Reset
}
```

4.1.4 Sending setting command

/**********/

The following example shows how to set the frequency of the 87237.

/**********/

```
void SimpleSettings()
{
    ViStatus iStatus;
    ViUInt32 uiRetCnt;
    //Set the frequency to 128MHz
    iStatus = viWrite(iDevHandle, "FREQ 1.2e8\n", strlen("FREQ 1.2e8\n"), &uiRetCnt);
}
```

4.1.5 Read the state of measuring instrument

```
/*****************************************************************************/
```

The following examples show how to read the set state of the instrument.

```
/*****************************************************************************/
void ReadSettings()
{
    ViStatus iStatus; //Status
    ViUInt32 uiRetCnt; //read the returned bytes
    char rgcBuf[256]; //Temporary buffer
    char* pcCmd = NULL; //Command pointer

    //QueryFreq
    pcCmd = "FREQ?\n";
    iStatus = viWrite(iDevHandle, pcCmd, strlen(pcCmd), &uiRetCnt);
    Sleep(10);
    iStatus = viRead(iDevHandle, rgcBuf, sizeof(rgcBuf), &uiRetCnt);
    //Print debugging information
    printf("frequency %s", rgcBuf);
}
```

4.1.6 Command synchronization

```
/*****************************************************************************/
```

The following examples illustrate the methods for command synchronization with sweep process.

```
/*****************************************************************************/
void SweepSync()
{
    ViStatus iStatus; //Status
    ViUInt32 uiRetCnt; //read the returned bytes
    ViEventType eType; //Event Type
    ViEvent eEvent; //Event
    int iStat; //State word
    char rgcOpcOk[256]; //OPC string
    char* pcCmd = NULL; //Command pointer

    /*************************************************************************/
    /* The command INITiate[:IMMediate] is used to start single sweep (when
    continuous sweep is OFF, INIT:CONT OFF)*/
    /* Only at the end of single sweep can the next command in the command
```

```
        buffer be executed                    */                    */
        /*****************************************************************/
        pcCmd = "INIT:CONT OFF\n";
        iStatus = viWrite(iDevHandle, " pcCmd ", strlen(pcCmd), &uiRetCnt);
        //Method 1 for waiting for the sweep to end: use *WAI
        pcCmd = "ABOR;INIT:IMM;*WAI\n";
        iStatus = viWrite(iDevHandle, " pcCmd ", strlen(pcCmd), &uiRetCnt);
        //Method 2 for waiting for the sweep to end: use *OPC?
        pcCmd = "ABOR;INIT:IMM; *OPC?\n";
        iStatus = viWrite(iDevHandle, " pcCmd ", strlen(pcCmd), &uiRetCnt);
        iStatus = viRead(iDevHandle, rgcOpcOk, 2, &uiRetCnt); //wait for *OPC to return
        "1"
        //Main program continues……
    }
```

4.2 Examples of advanced operation

- Examples of USBTMC programmed control 158

4.2.1 Examples of USBTMC programmed control

- Before Using the Examples.....158
- Example.....158

4.2.1.1 Before using the examples

If you use VISA library for programmed control, you need to install VISA library, such as KeysightIO17.2, NI VISA5.0, etc.
Note that lower versions of the VISA library do not support the USBTMC programming feature.

4.2.1.2 Examples

- Using VISA library and C++ language to implement web.....158

1) Using VISA library and C++ language to implement web programs

/**********

In this example, USB programmed control is implemented via the VISA library.
Start VS2005 or above, add the necessary files and add the directory where visa.h is located to the "Include files" path of the project (to add: press "Tools" in the menu, then

press "Options", "Projects and Solutions", "VC++ Directory", and then select "Include files" under the "Show directories with the following content", and add the directory where visa.h is located in the list. (For the English version of VS, please refer to the above method to set up, without further explanation.)

Enter the following code into your .cpp file

```
**********************************************************/
#include "stdafx.h"
#include<visa.h>
#include <stdio.h>
#include <stdlib.h>

#define M_USB_VID     0x3399       //USB Vendor ID
#define M_USB_PID     0x3830       //Instrument ID
#define M_USB_SN      "ZGK00541" //Instrument Serial Number (No.)

//Description:
ViSession g_uiDefaultRm = 0; //Resource Handle
ViSession g_uiInstrHandle = 0; //Device VISA Handle

void UsbInit(void);
void UsbTest(void);
void UsbClose(void);
//USB test main program
void UsbTestMain(void)
{
    //Initialize (user opens the instrument connection before using the instrument)
    UsbInit();

    //Test (user can communicate with the instrument)
    UsbTest();

    //OFF (close the instrument connection when the user is no longer using the
instrument)
    UsbClose();
}

//USB initialization
void UsbInit(void)
{
    ViStatus iStatus = 0; //state
    ViChar rgcBuf[256]; //temporary buffer
```

```
    sprintf(rgcBuf, "USB0::%d::%d::%s::0::INSTR", M_USB_VID,M_USB_PID,M_USB_SN);
    iStatus = viOpenDefaultRM(&g_uiDefaultRm); //open VISA task
    if (iState)
    {
        printf("The task cannot be opened. Please recheck the device and connect\n");
    }
    else
    {
        iStatus = viOpen(g_uiDefaultRm, rgcBuf, VI_NULL, 5000, &g_uiInstrHandle);
        if (iState)
        {
            printf("The devicee cannot be opened. Please recheck the device and
connect\n");
        }
        else
        {
        }
    }
}
//USB Test
void UsbTest(void)
{
    ViStatus iStatus = 0; //state
    ViChar rgcBuf[256]; //temporary buffer
    ViByte rgcRead[256]; //Read buffer
    ViReal64 rgdFreq[2]; //Frequency array
    ViUInt32 uiRetCnt; //Return length

    if (0 == g_uiInstrHandle)
    {
        printf("The device is not opened. Please recheck the device and connect\n");
    }
    else
    {
        //1) First query the frequency of channel A, and store it in rgdFreq[0]
        viPrintf(g_uiInstrHandle, "SENS1:FREQ?\n");
        Sleep(10);
        viRead(g_uiInstrHandle, rgcRead, sizeof(rgcRead), &uiRetCnt);
        rgcRead[uiRetCnt] = 0;
        printf((PCHAR)rgcRead);

        sscanf((PCHAR)&rgcRead[0], "%lf", &rgdFreq[0]);
```

```
            sprintf(rgcBuf, "Frequency of channel A: %lg\n", rgdFreq[0]);
            printf(rgcBuf);

            //2) Set the frequency of channel A to .78GHz
            viPrintf(g_uiInstrHandle, "SENS1:FREQ    %lfGHz\n", 16.78);

            //3) Query the frequency of channel A, and store it in rgdFreq[1]
            viPrintf(g_uiInstrHandle, "SENS1:FREQ?\n");
            viScanf(g_uiInstrHandle, "%t", rgcBuf);            //Put the query results into an
array
            printf(rgcBuf);

            sscanf(rgcBuf, "%lf", &rgdFreq[1]);
            sprintf(rgcBuf, "Frequency of channel A: %lg\n", rgdFreq[1]);
            printf(rgcBuf);

            //4) Restore the frequency of channel A
            viPrintf(g_uiInstrHandle, "SENS1:FREQ    %lg\n", rgdFreq[0]);
        }
}
//USB close
void UsbClose(void)
{
        if (0 != g_uiInstrHandle)
        {
            viClose(g_uiInstrHandle); //Close the device
            g_uiInstrHandle = 0;
        }
        if (0 != g_uiDefaultRm)
        {
            viClose(g_uiDefaultRm); //Close default tasks
            g_uiDefaultRm = 0;
        }
}
```

4.2.2 Linux development examples

Currently, there are many versions of Linux systems in the market, and most of them cannot install the VISA library. The example in this section uses the system common read/write command to realize the programmable use of the 87237.
Note: Linux shall support the USBTMC protocol.

4.2 Examples of advanced operation

```
*************************************************************************
#include <fcntl.h>

#define O_RDWR                  2
#define USB_MAX_BUFFER     4096
#define M_CMD_IDN             "*IDN?"
#define M_CMD_MEASURE       "MEAS?"

int UsbtmcTest()
{
      //If there are multiple devices, it may be "/dev/usbtmc1",.....
      char* pcUsbtmcDev = "/dev/usbtmc0";
      char* pcWritePtr;
      unsigned int uiHandle;
      unsigned long dwReadCnt;
      char cReadBuf[USB_MAX_BUFFER];
      unsigned int iStatus;

      //The device must be turned on before use
      uiHandle = open(pcUsbtmcDev, O_RDWR, 666);

      //Send the command "*IDN?" to the device
      pcWritePtr = M_CMD_IDN;
      iStatus = write(uiHandle, pcWritePtr, strlen(pcWritePtr));
      //Read information from the device
      dwReadCnt = read(uiHandle, cReadBuf, USB_MAX_BUFFER);
      printf("IDN is: %s", cReadBuf);

      //Send the command "MEAS?" to the device
      pcWritePtr = M_CMD_MEASURE;
      iStatus = write(uiHandle, pcWritePtr, strlen(pcWritePtr));
      //Read information from the device
      dwReadCnt = read(uiHandle, cReadBuf, USB_MAX_BUFFER);
      printf("The measurement result is : %s", cReadBuf);

      //The device must be turned off at the end of use
      close(uiHandle);
}
*************************************************************************
```

4.3 Application Examples

4.3.1 External trigger buffer measurement

```
TRIG:SOUR EXT                    //Trigger source set to external
SENS:FREQ 1GHz                   //Set the frequency to 1GHz
SENS:FREQ:STEP 0                 //Sweep points 0
SENS:BUFF:COUN 10                //Number of buffer measurements 10
TRIG:DELAY 100us          // Capture external trigger signal with 100us delay
SENS:SWE:APER 1ms         //Aperture set to 1ms
Repeat the following commands to obtain multiple measurement results
INIT:CONT ON                     //Initiate measurement
*OPC                             //Set the operation end bit in the standard
event state register
*ESR?                            //Query the value of the standard event state
register
FETC?                            //Fetch measurement results
```

4.3.2 External trigger buffer sweep frequency measurement

```
TRIG:SOUR EXT                    //Trigger source set to external
SENS:FREQ:STAR 1GHz              //Set the start frequency to 1GHz
SENS:FREQ:STOP 10GHz             //Set start frequency to 10GHz
SENS:FREQ:STEP 10                //Sweep points 10
TRIG:DELAY 100us          // Capture external trigger signal with 100us delay
SENS:SWE:APER 1ms         //Aperture set to 1ms
Repeat the following commands to obtain multiple measurement results
INIT:CONT ON                     //Initiate measurement
*OPC                             //Set the operation end bit in the standard
event state register
*ESR?                            //Query the value of the standard event state
register
FETC?                            //Fetch measurement results
```

4.3.3 Fast measurement mode (free run)

"AVER" mode (1000 readings/sec)

```
SYST:PRES                  //Reset the device to its default value
SENS:FREQ 1GHz             //Set the frequency to 1GHz
UNIT:POW W                 //Calculation 1 power measurement unit W
```

```
SENS:AVER:SDET OFF          //Step detection OFF
SENS:AVER OFF               //Measurement average OFF
SENS:MRAT FAST              //Measurement speed in fast mode
TRIG:COUN 200               //Measurement buffer size 200
SENS:SWE:APER 1e-3          //Set the aperture length to 1ms
FETC?                       //Fetch measurement results
```

4.3.4 Fast measurement mode (external trigger)

When the aperture length is not larger than the time interval of the external trigger signal, the external trigger signal can be captured continuously without interruption.

The measurement speed is determined by the greater of the time interval of the external trigger signal and the length of the aperture, with max 1000 readings/second.

The following routines help users establish GSM timeslot measurement based on external trigger (16 GSM Architecture), which can continuously measure 128 timeslots.

```
SYST:PRES                   //Reset the device to its default value
SENS:FREQ 1GHz              //Set the frequency to 1GHz
TRIG:SOUR EXT               //Set to external trigger
SENS:AVER:SDET OFF          //Step detection OFF
SENS:AVER OFF               //Measurement average OFF
SENS:MRAT FAST              //Measurement speed in fast mode
TRIG:COUN 128               //Measurement buffer size 128
TRIG:DEL 50e-6              //Set the trigger delay to 50us
SENS:SWE:APER 1e-3          //Set the aperture length to 1ms
FETC?                       //Fetch measurement results
```

4.3.5 Continuous and uninterrupted measurements

The average power of known periodic signals, especially large periodic signals, can be accurately measured.

Mode 1: Accurate measurement of the signal with a period of 5s = 50ms*100 (FETCH mode)

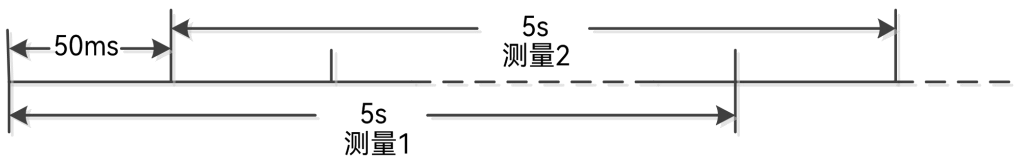


Figure 4.1 Measurement process

```
SYST:PRES                   //Reset the device to its default value
SENS:FREQ 1GHz              //Set the frequency to 1GHz
CAL:ZERO:AUTO OFF           //Auto-zeroing OFF
```

```
CAL:AUTO OFF                    //Auto-calibration OFF
SENS:AVER:SDET OFF              //Step detection OFF
SENS:SWE:APER 50e-3             //Set the aperture measurement length to
50ms
SENS:AVER:COUN 100              //Set the average number of measurements
to 100
// Delay 5s
FETC?                           //Fetch measurement results
```

Mode 2: Accurate measurement of the signal with a period of 5s = 50ms*100 (MEAS mode)

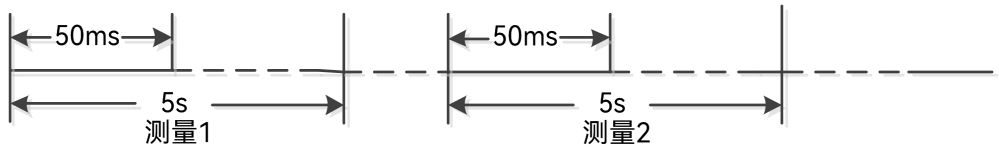


Figure 4.2 Measurement process

```
SYST:PRES                       //Reset the device to its default value
SENS:FREQ 1GHz                  //Set the frequency to 1GHz
CAL:ZERO:AUTO OFF               //Auto-zeroing OFF
CAL:AUTO OFF                    //Auto-calibration OFF
SENS:AVER:SDET OFF              //Step detection OFF
SENS:SWE:APER 50e-3             //Set the aperture measurement length to
50ms
SENS:AVER:COUN 100              //Set the average number of measurements
to 100
MEAS?                           //Perform a measurement and obtain the
results
```

Mode 3: Accurate measurement of the signal with a period of 1s = 50ms*20 (The measurement period of this method cannot exceed 1 second)

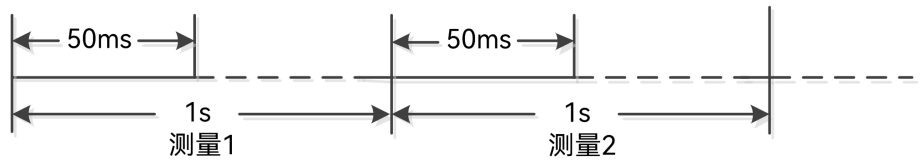


Figure 4.3 Measurement process

```
SYST:PRES                       //Reset the device to its default value
SENS:FREQ 1GHz                  //Set the frequency to 1GHz
CAL:ZERO:AUTO OFF               //Auto-zeroing OFF
CAL:AUTO OFF                    //Auto-calibration OFF
```

4.3 Application Examples

SENS:AVER OFF //Measurement average OFF
SENS:AVER:SDET OFF //Step detection OFF
SENS:MRAT FAST //Measurement speed in fast mode
TRIG:COUN 20 //Measurement buffer size 20
SENS:SWE:APER 50e-3 //Set the aperture measurement length to 50ms
FETC? //Obtain the measurement results (the average power in this period can be obtained by summing and averaging all the measurement results)

5. Error description

This chapter will show you how to find problems and accept after-sales service, and explain error message of the 87237.

Error message..... 167
Method to obtain after-sales services..... 172

5.1 Error message

The 87237 records errors during measurement in two ways: error message queues displayed on the front panel and SCPI (in the remote control mode) error message queues. which are stored and managed separately.

1) Error message format and description

In remote control mode, errors are recorded in the error/event queue of the state reporting system, and can be queried with the command "SYSTem:ERRor?". The format is as follows:

"<Error code>, "<Error in error queue>; ["Detailed error description]"

Example:

"-110,"Data out of range;"

The programmed control error message includes two types:

A negative error code defined by the SCPI standard. This type of error is not specified here.

Table 5.1 List of descriptions on instrument feature error messages

Failure code	Error description
-101	· Invalid character: There are invalid characters in the command string (command o parameter). For example: AVER:COUN !6
-102	· Syntax error: The command string syntax is invalid. For example: DISPlay:ACT, CH1
-108	· Parameters are not allowed: the command has too many parameters, or the commanc without parameters follows the parameters. For example: TRAC:AUT ON
-109	· Missing parameters: The command has too few parameters.

Failure code	Error description
	For example: AVER
-112	. Command string is too long: A single segment of the command has more than 12 characters. For example: OUTPutROSCillatorSTATe ON
-113	Undefined header Undefined header: 87237 receives an unrecognized command. Possible causes: Wrong spelling, or wrong abbreviation of the command, etc. For example: CALI:AUTO
-121	. Invalid characters in the numeric value: there are invalid characters in the numeric parameter. For example:SENS:CORR:GAIN2 #12
-123	. The index is too large: the index of numerical parameters exceeds 32000. For example: SENS:CORR:GAIN2 1E32001
-124	. Too many bits: the number of bits of numeric parameters exceeds 255, without the leading 0.
-128	. Numeric parameters are not allowed: commands that cannot receive numeric parameters receive a numeric value.
-131	. Invalid suffix: the suffix of numeric parameter is incorrect. For example: FREQ 10GZ
-134	. Suffix too long: the suffix exceeds 12 characters. For example: FREQ 10GHHHHHHHHHHHHHHHHHZ
-138	. Suffix not allowed: numeric parameters cannot be followed by suffixes. For example: SENS:CORR:GAIN2 12HZ
-148	. Character data is not allowed: check whether quotation marks need to be added. For example: MEM:CLE State_1 Correct: MEM:CLE "State_1"
-151	. Invalid string data: check whether the single quotation marks or double quotation marks of the string match. For example: MEM:CLE “State1

Failure code	Error description
-158	. Character data is not allowed: check whether the parameter type is valid. For example: OUTP:ROSC "ON"
-161	. Invalid block data: Check according to Section 7.7.6 of IEEE 488.2.
-168	. Block data not allowed: a valid data block was detected, but the command does no support data blocks. For example: OUTP:ROSC #15FETC?
-178	. Expression data not allowed: a valid expression was detected, but expression is no allowed in the 87237. For example: SENS:CORR:GAIN2 (1+3)
-211	Trigger ignored Trigger ignored: when 87237 is not in "wait for trigger" state, trig:imm, *trg and othe commands are received.
-213	Init ignored Initialization is ignored: when 87237 has been initialized, the measurement initializatior command is received. For example: INIT:CONT ON INIT
-214	Trigger deadlock Trigger deadlock
-220	Parameter error; Frequency list must be in ascending order. Parameter error; the frequency list must be arranged in ascending order
-221	. Setting conflict: there are many reasons for conflict, such as setting trigger delay during statistical measurement.
-222	. Data out of range: Numerical data is not within the valid range. for example: AVER:COUN 100000
-224	. Illegal parameter value: a discrete parameter was received, but it is invalid for thi command.
-226	Lists not same length List length is different
-230	Data corrupt or stale; Please calibrate Invalid or damaged data; Please calibrate.
-231	Data questionable; CAL ERROR

5.2 Method to obtain after-sales services

Failure code	Error description
	Data question; Calibration error: the 87237 calibration failed. The most likely reason is tha 87237 is not connected to the output of the calibration source during calibration. Data questionable; CAL ERROR Ch1 Data question; channel A calibration error: the 87237 calibration failed. The most likely reason is that 87237 is not connected to the output of the calibration source during calibration. Data questionable; CAL ERROR Ch2 Data question; channel B calibration error: the 87237 calibration failed. The most likely reason is that 87237 is not connected to the output of the calibration source during calibration. Data questionable; Input Overload Data question; Input overload: the power input exceeds the upper power limit of the 87237. Data questionable; Input Overload Ch1 Data question; Input overload: the power input exceeds the upper power limit of the 87237. Data questionable; Lower window log error Data question; Logarithm error in lower measurement: when the difference measuremen is carried out, the measurement result is 0 and the unit of display is logarithm. Data questionable; Upper window log error Data question; Logarithm error in upper measurement: when the difference measuremen is carried out, the measurement result is 0 and the unit of display is logarithm. Data questionable; ZERO ERROR Data question; Zeroing error: the 87237 zeroing failed. The most likely reason is that there is power signal input during zeroing. Data questionable; ZERO ERROR Ch1 Data question; channel A zeroing error: the 87237 zeroing failed. The most likely reason is that there is power signal input during zeroing.
-241	. Missing hardware: 87237 cannot execute the command, type mismatch.
-310	System error; Sensor EEPROM Read Failed - critical data not found or unreadable System error; EEPROM read failed - key data not found or unreadable System error; Sensor EEPROM Read Failed - unknown EEPROM table format System error; EEPROM read failed - unknown EEPROM format.
-321	. Exceeding memory
-330	Self-test Failed; Self-test failed
-350	Queue overflow

Failure code	Error description
	Queue overflow: the error queue is full, and subsequent errors will no longer be recorded.
-410	Query INTERRUPTED The query is interrupted: A command needs to send data to the output buffer, but the senc buffer already contains the data sent by the previous command (without overwriting the previous data). The output buffer is cleared on shutdown or on receipt of the *RST command. Refer to Section 6.3.2.3 of IEEE 488.2 for details.
-420	Query UNTERMINATED Query not ended: The 87237 is set to Speak (i.e., send data to the interface bus), but the command to send data to the output buffer is not received. Refer to Section 6.3.2.2 of IEEE 488.2 for details.
-430	Query DEADLOCKED Query deadlock: The output buffer cannot hold too much data generated by the command, and the output buffer is full. Command execution continues but data is lost Refer to Section 6.3.1.7 of IEEE 488.2 for details.
-440	Query UNTERMINATED after indefinite response Query not ended after an indefinite response: Some combination queries may generate illegal response messages. If a query command that produces an indefinite response (any block response of indefinite length or any ASCII response data) is not the last query command, the instrument will report a query error and no further responses will be sen after that query command. Refer to Section 6.5.7.5 of IEEE 488.2 for details.

2) Error message type

The error event corresponds only to one type of error message, and the error message types are introduced in details below:

- Query error (−499 to −400): indicates the output queue control of the instrument detects the message exchange protocol error specified in Chapter 6 of IEEE 488.2. At this time, the event state register of the error query bit (bit2) is set (for details, refer to 6.5 of IEEE 488.2). The data cannot be successfully read from the output queue at this time.
- Instrument characteristic error (-399 to -300, 201 to 703, and 800 to 810): indicating that the instrument operation is not successful, and the reason may be abnormal hardware or firmware state. Such error codes are often used self-detection of the instrument. At this point, the instrument characteristic error bit (bit3) of the event state register is set.
- Execution error (-299 to -200): indicating that an error is detected during the measurement of the instrument. At this point, the execution error bit (bit4) of the event state register is set.
- Command error (-199 to -100): indicating a syntax error detected during command parsing of the instrument, usually due to an incorrect command

format. At this point, the command error bit (bit5) of the event state register is set.

5.2 Method to obtain after-sales services

- Contacting us.....172
- Package and mailing.....172

5.2.1 Contacting us

In case of any failure to the 87237, check and save the error message, analyze possible causes, and refer to the methods provided in "6.2 Fault diagnosis and troubleshooting" in the user's manual for preliminary troubleshooting. If the problem cannot be solved, contact the service and consultation center of the Company as per the contact information provided below and provide us with the error message collected. We will coordinate with you to solve the problem as soon as possible.

Contact information:

Service consultation	400-1684191
Website:	www.ceyear.com
Email:	techbb@ceyear.com
Postal code:	266555
Address:	No. 98, Xiangjiang Road, Huangdao District, Qingdao, Shandong Province, China

5.2.2 Package and mailing

In case of any failure to the 87237 that is difficult to be eliminated, contact us by phone or fax. If it is confirmed that the 87237 has to be returned for repairing, pack it with the original packing materials and case by following the steps below:

1) Prepare a detailed description of the failure of the 87237 and put it into the package along with it.
2) Pack the 87237 with the original packing materials, so as to minimize possible damage;
3) Place cushions at the four corners of the outer packing carton, and place the network analyzer in the outer packing carton.
4) Seal the opening of the packing carton with adhesive tape and reinforce the packing carton with nylon tape.
5) Specify text like "Fragile"! Do not touch! Handel with care!" and so on.
6) Please consign it as precision instruments.
7) Keep a copy of all shipping documents.

NOTICE

Precautions on packing the 87237

Using other materials for packing the 87237 may damage the instrument. Never use polystyrene beads as packing materials because on the one hand, they cannot provide sufficient protection on the instrument, and on the other hand, they can be sucked in to the instrument fan by the static electricity generated, resulting in instrument damage.

TIPS

Instrument package and transportation

Please follow carefully the precautions described in "3.1.1.1 Unpacking" of the User's manual when transporting or handling the instrument (for example, damage occurred during delivery).

Annex

- Annex A Zoom table of SCPI classified by subsystem.....175

Annex A Zoom table of SCPI classified by subsystem

Table 1 Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
44	*CLS	Clear the instrument state data structure
44	*DDT	Query or set the operation in response to *TRG general command. Note: The 87237 does not support this command at the moment.
45	*ESE	Query or set the standard event state enable register
46	*ESR?	Query the value of the standard event state register
46	*IDN?	Query the identification string of 87237
46	*OPC	When all waiting operations are completed, set the operation end bit in the standard event state register
47	*OPT?	Query the instrument option configuration.
47	*RCL	Call the state in the specified storage call register
47	*RST	Reset the 87237
47	*SAV	Store the instrument state into the specified register
48	*SRE	Query or set the service request register
48	*STB?	Query the state word.
49	*TRG	Trigger all channels waiting for trigger
49	*TST?	Perform self-test

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
49	*WAI	Place the 87237 in the wait state
143	:ABORt[1]	Stop the measurement of the corresponding channel of the 87237
51	:CALCulate[1]|2|3|4:FEED[1]|2	Query or set the measurement mode of measurement display
52	:CALCulate[1]|2|3|4:GAIN[:MAGNitude]	Query or set the display offset value in the specified measurement
52	:CALCulate[1]|2|3|4:GAIN:STATe	Query or set the ON/OFF state of the display offset
53	:CALCulate[1]|2|3|4:LIMit:CLEar:AUTO	Control when the limit FCO (failure count) is cleared
53	:CALCulate[1]|2|3|4:LIMit:CLEar[:IMMediate]	Clear the FCO (failure count) of the specified measurement
54	:CALCulate[1]|2|3|4:LIMit:FAIL?	Query whether the specified measurement exceeds the limit.
54	:CALCulate[1]|2|3|4:LIMit:FCOunt?	Query the limit detection failure count (FCO) of the specified measurement
54	:CALCulate[1]|2|3|4:LIMit:LOWer[:DATA]	Query or set the lower limit of the specified measurement limit
56	:CALCulate[1]|2|3|4:LIMit:STATe	Query or set the ON/OFF state of the specified measurement limit detection
56	:CALCulate[1]|2|3|4:LIMit:UPPer[:DATA]	Query or set the upper limit of the specified measurement limit
57	:CALCulate[1]|2|3|4:MATH[:EXPRession]	Query or set the specified measurement expression
58	:CALCulate[1]|2|3|4:MATH[:EXPRession]:CATalogue?	Enumerate all measurement expressions
58	:CALCulate[1]|2|3|4:RELative[:MAGNitude]:AUTO	Set the reference value for relative measurement
59	:CALCulate[1]|2|3|4:RELative[:MAGNitude]:VALue?	Query the reference value for relative measurement

Index	Command	Brief description of functions
59	:CALCulate[1]|2|3|4:RELative:STATe	Query or set the relative measurement ON/OFF state
60	:CALibration[1][:ALL]	Zeroing and calibration of the 87237
60	:CALibration[1]:AUTO	Calibration of the 87237
61	:CALibration[1]:ZERO:AUTO	Zeroing of the 87237
71	:DISPlay[:WINDow[1]|2][:NUMeric[1]|2]:RESolution	Query or set the display resolution of the specified measurement
62	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]?	Set the specified measurement as absolute power measurement, turn off relative measurement, and return the measurement value
63	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?	Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value
63	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?	Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value
63	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?	Set the power measurement mode of the specified measurement as ratio measurement, turn off relative measurement, and return the measurement value
64	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?	Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value
64	:FETCh[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?	Set the specified measurement as absolute power measurement, turn on relative measurement, and

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
		return the measurement value
71	:FORMat[:READings]:BORDer	Query or set the transmission order of binary data
72	:FORMat[:READings][:DATA]	Query or set the data transmission format
143	:INITiate[1]:CONTinuous	Query or set the trigger state
143	:INITiate:CONTinuous:ALL	Query or set the trigger state of all channels
144	:INITiate:CONTinuous:SEQuence[1]	Query or set the trigger state
144	:INITiate[1][:IMMediate]	Set to "wait for trigger" state
145	:INITiate[:IMMediate]:ALL	Set all channels to "wait for trigger" state
145	:INITiate[:IMMediate]:SEQuence[1]	Set to "wait for trigger" state
145	:INPut:TRIGger:IMPedance	Query or set the trigger input impedance. Note: The 87237 does not support this command at the moment. This command is used for expansion.
65	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]?	Set the specified measurement as absolute power measurement, turn off relative measurement, and return the measurement value
66	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?	Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value
66	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?	Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value
66	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?	Set the power measurement mode of the specified measurement as ratio measurement, turn off relative measurement, and return the measurement value

Index	Command	Brief description of functions
67	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?	Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value
67	:MEASure[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?	Set the specified measurement as absolute power measurement, turn on relative measurement, and return the measurement value
73	:MEMory:CATalog[:ALL]?	Enumerate the user configurations in the 87237, including storage call configuration, frequency response offset table (FDO), etc
73	:MEMory:CATalog:STATe?	Enumerate the storage call configuration in the 87237
74	:MEMory:CATalog:TABLe?	Enumerate the frequency response offset table in the 87237
74	:MEMory:CLEar[:NAME]	It is used to clear the frequency response offset table or storage call table specified in the power
75	:MEMory:CLEar:TABLe[1]|2|3|4|5|6|7|8|9|10	Clear the specified frequency response offset table
75	:MEMory:FREE[:ALL]?	Query the total number of bytes unused in the user configuration space and the number of bytes
75	:MEMory:FREE:STATe?	Query the total number of bytes unused in the storage call space and the number of bytes
76	:MEMory:FREE:TABLe?	Query the total number of unused bytes and the number of bytes used in the frequency response offset table

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
76	:MEMory:NSTates?	Query the number of storage call states, and always return 10
76	:MEMory:STATe:CATalog?	Enumerate the names of all storage call states
77	:MEMory:STATe:DEFine	Query or set the name of the storage call state register.
77	:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:DEFine	Query or set the name in the specified frequency response offset table
77	:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency	Query or set the frequency list in the specified frequency response offset table
78	:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:FREQuency:POINts?	Query the frequency points in the specified frequency response offset table
78	:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]	Query or set the amplitude offset list in the specified frequency response offset table
79	:MEMory:TABLe[1]|2|3|4|5|6|7|8|9|10:GAIN[:MAGNitude]:POINts?	Query the amplitude offset points in the specified frequency response offset table
79	:MEMory:TABLe:MOVE	Rename the specified frequency response offset table
80	:MEMory:TABLe:SELect	Query or set the current frequency response offset table
68	:READ[1]|2|3|4[:SCALar][:POWer][:AC]?	Set the specified measurement as absolute power measurement, turn off relative measurement, and return the measurement value
68	:READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence?	Set the specified measurement as differential power measurement, turn off relative measurement, and return the measurement value

Index	Command	Brief description of functions
69	:READ[1]|2|3|4[:SCALar][:POWer][:AC]:DIFFerence:RELative?	Set the specified measurement as differential power measurement, turn on relative measurement, and return the measurement value
69	:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio?	Set the power measurement mode of the specified measurement as ratio measurement, turn off relative measurement, and return the measurement value
69	:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RATio:RELative?	Set the power measurement mode of the specified measurement as ratio measurement, turn on relative measurement, and return the measurement value
70	:READ[1]|2|3|4[:SCALar][:POWer][:AC]:RELative?	Set the specified measurement as absolute power measurement, turn on relative measurement, and return the measurement value
81	[:SENSe[1]]:AVERage:COUNt	Query and set the average times of channel measurement
82	[:SENSe[1]]:AVERage:COUNt:AUTO	Query or set the channel measurement auto-average state
82	[:SENSe[1]]:AVERage:RESet	Clear the averaging buffer and restart averaging.
82	[:SENSe[1]]:AVERage:SDETect	Query or set the channel step detection state
83	[:SENSe[1]]:AVERage[:STATe]	Query or set the ON/OFF state of channel measurement average
83	[:SENSe[1]]:BUFFer:COUNt	Query or set the buffer size of external trigger measurement, which is only used for external trigger measurement.
84	[:SENSe[1]]:CORRection:CSET2:STATe	Query or set the activation

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
		state of the frequency response offset table
84	[:SENSe[1]]:CORRection:DCYCle[:INPut][:MAGNitude]	Query or set the channel duty cycle setting value for pulse power measurement.
85	[:SENSe[1]]:CORRection:DCYCle[:INPut]:STATe	Query or set the ON/OFF state of channel duty cycle for pulse power measurement.
86	[:SENSe[1]]:CORRection:FDOFfset[:INPut][:MAGNitude]?	Query the frequency response offset factor.
86	[:SENSe[1]]:CORRection:GAIN[1]|2|3|4[:INPut][:MAGNitude]	Set or query four kinds of gains for the current device
87	[:SENSe[1]]:CORRection:GAIN[1]|2|3|4[:INPut]:STATe	Set or query the ON/OFF state of four kinds of gains for the current device
87	[:SENSe[1]]:FREQuency[:CW|FIXed]	Query or set the frequency.
88	[:SENSe[1]]:FREQuency[:CW|FIXed]:STARt	Query or set the frequency for the start frequency of the external trigger buffer sweep measurement.
88	[:SENSe[1]]:FREQuency[:CW|FIXed]:STEP	Query or set the frequency step number for the external trigger buffer sweep measurement.
89	[:SENSe[1]]:FREQuency[:CW|FIXed]:STOP	Query or set the frequency for the stop frequency of the external trigger buffer sweep measurement.
90	[:SENSe[1]]:MRATe	Query or set the measurement speed.
90	[:SENSe[1]]:POWer:AC:RANGe	Query and set the current range.
91	[:SENSe[1]]:POWer:AC:RANGe:AUTO	Query whether auto range is enabled and make corresponding settings.
91	[:SENSe[1]]:SWEep[1]|2|3|4:AUTO	Query or set the automatic gate state of the specified gate
92	[:SENSe[1]]:SWEep:APERture	Query or set the measurement aperture or

Index	Command	Brief description of functions
		measurement time interval.
150	:SERVice:SENSor[1]:CDATe?	Query the calibration date of the 87237
150	:SERVice:SENSor[1]:CPLace?	Query the calibration location of the 87237
150	:SERVice:SENSor[1]:FREQuency:MAXimum?	Query the maximum frequency of the 87237
150	:SERVice:SENSor[1]:FREQuency:MINimum?	Query the minimum frequency of the 87237
151	:SERVice:SENSor[1]:SNUMber?	Query the serial number of the 87237
151	:SERVice:SENSor[1]:TYPE?	Query the type of the 87237
151	:SERVice:SNUMber	Query or set the serial number of the 87237
115	:STATus:DEVice:CONDition?	Query the value in the device state condition register
116	:STATus:DEVice:ENABle	Query or set the device state event enable register
116	:STATus:DEVice[:EVENt]?	Query the device event register
116	:STATus:DEVice:NTRansition	Query or set the negative transition filter of the device
117	:STATus:DEVice:PTRansition	Query or set the positive transition filter of the device
117	:STATus:OPERation:CALibrating[:SUMMary]:CONDition?	Query the value in the calibration operation state condition register
118	:STATus:OPERation:CALibrating[:SUMMary]:ENABle	Query or set the calibration operation event enable register
118	:STATus:OPERation:CALibrating[:SUMMary][:EVENt]?	Query the calibration operation event register
118	:STATus:OPERation:CALibrating[:SUMMary]:NTRansition	Query or set the negative transition filter for calibration operation
119	:STATus:OPERation:CALibrating[:SUMMary]:PTRansition	Query or set the positive transition filter for calibration operation
119	:STATus:OPERation:CONDition?	Query the value in the operation state condition

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
		register
120	:STATus:OPERation:ENABle	Query the value in the operation state condition register
120	:STATus:OPERation[:EVENt]?	Query the operation state event register
121	:STATus:OPERation:LLFail[:SUMMary]:CONDition?	Query the value in the lower limit detection operation state condition register
121	:STATus:OPERation:LLFail[:SUMMary]:ENABle	Query or set the event enable register for lower limit detection operation
121	:STATus:OPERation:LLFail[:SUMMary][:EVENt]?	Query the event register for lower limit detection operation
122	:STATus:OPERation:LLFail[:SUMMary]:NTRansition	Query or set the negative transition filter for lower limit detection operation
122	:STATus:OPERation:LLFail[:SUMMary]:PTRansition	Query or set the positive transition filter for lower limit detection operation
122	:STATus:OPERation:NTRansition	Query or set the negative transition filter for operation state
123	:STATus:OPERation:PTRansition	Query or set the positive transition filter for operation state
123	:STATus:OPERation:SENSe[:SUMMary]:CONDition?	Query the value in the sensor operation state condition register
124	:STATus:OPERation:SENSe[:SUMMary]:ENABle	Query or set the sensor operation event enable register
124	:STATus:OPERation:SENSe[:SUMMary][:EVENt]?	Query the event register for sensing operation
124	:STATus:OPERation:SENSe[:SUMMary]:NTRansition	Query or set the negative transition filter for sensing operation
125	:STATus:OPERation:SENSe[:SUMMary]:PTRansition	Query or set the positive transition filter for sensing

Index	Command	Brief description of functions
		operation
125	:STATus:OPERation:TRIGger[:SUMMary]:CONDition?	Query the value in the trigger operation state condition register
126	:STATus:OPERation:TRIGger[:SUMMary]:ENABle	Query or set the trigger operation event enable register
126	:STATus:OPERation:TRIGger[:SUMMary][:EVENt]?	Query the event register for trigger operation
126	:STATus:OPERation:TRIGger[:SUMMary]:NTRansition	Query or set the negative transition filter for trigger operation
127	:STATus:OPERation:TRIGger[:SUMMary]:PTRansition	Query or set the positive transition filter for trigger operation
127	:STATus:OPERation:ULFail[:SUMMary]:CONDition?	Query the value in the upper limit detection operation state condition register
128	:STATus:OPERation:ULFail[:SUMMary]:ENABle	Query or set the event enable register for upper limit detection operation
128	:STATus:OPERation:ULFail[:SUMMary][:EVENt]?	Query the event register for upper limit detection operation
128	:STATus:OPERation:ULFail[:SUMMary]:NTRansition	Query or set the negative transition filter for upper limit detection operation
129	:STATus:OPERation:ULFail[:SUMMary]:PTRansition	Query or set the positive transition filter for upper limit detection operation
129	:STATus:PRESet	Preset some state registers
130	:STATus:QUEStionable:CALibration[:SUMMary]:CONDition?	Query the value in the calibration questionable state condition register
130	:STATus:QUEStionable:CALibration[:SUMMary]:ENABle	Query or set the calibration questionable event enable register
131	:STATus:QUEStionable:CALibration[:SUMMary][:EVENt]?	Query the calibration operation event register
131	:STATus:QUEStionable:CALibration[:SUMMary]:NTR	Query or set the questionable

Annex A Zoom table of SCPI classified by subsystem

Index	Command	Brief description of functions
	ansition	negative transition filter for calibration operation
132	:STATus:QUEStionable:CALibration[:SUMMary]:PTRansition	Query or set the questionable positive transition filter for calibration operation
132	:STATus:QUEStionable:CONDition?	Query the value in the questionable state condition register
132	:STATus:QUEStionable:ENABle	Query or set the questionable state event enable register
133	:STATus:QUEStionable[:EVENt]?	Query the questionable state event register
133	:STATus:QUEStionable:NTRansition	Query or set the questionable negative transition filter for operation state
134	:STATus:QUEStionable:POWer[:SUMMary]:CONDition?	Query the value in the power questionable state condition register
134	:STATus:QUEStionable:POWer[:SUMMary]:ENABle	Query or set the power questionable event enable register
134	:STATus:QUEStionable:POWer[:SUMMary][:EVENt]?	Query the questionable event register for power operation
135	:STATus:QUEStionable:POWer[:SUMMary]:NTRansition	Query or set the questionable negative transition filter for power operation
135	:STATus:QUEStionable:POWer[:SUMMary]:PTRansition	Query or set the questionable positive transition filter for power operation
136	:STATus:QUEStionable:PTRansition	Query or set the questionable positive transition filter for operation state
138	:SYSTem:ERRor:CODE?	Return error code from error queue
138	:SYSTem:ERRor[:NEXT]?	Return error code and error information from the error queue
139	:SYSTem:HELP:HEADers?	Query the list of commands supported by 87237
139	:SYSTem:IDN	Query or set "*IDN?" of the

Index	Command	Brief description of functions
		87237 Query the returned user-defined string.
139	:SYSTem:IDN:AUTO	Query or set the "*IDN?" of the 87237 Whether to return the switch of the user-defined string.
140	:SYSTem:PRESet	Reset 87237 to the state specified by the parameter
142	:SYSTem:VERSion?	Query the SCPI version number used by 87237
145	:TRIGger[1][:IMMediate]	Set to "wait for trigger" state
146	:TRIGger[:SEQuence[1]]:COUNt	Query or set the number of trigger event detection/measurement cycles.
146	:TRIGger[:SEQuence[1]]:DELay	Query or set the trigger delay
147	:TRIGger[:SEQuence[1]]:IMMediate	Set to "wait for trigger" state
147	:TRIGger[:SEQuence[1]]:LEVel	Query or set the trigger level
147	:TRIGger[:SEQuence[1]]:SLOPe	Query or set the trigger type
148	:TRIGger[:SEQuence[1]]:SOURce	Query or set the trigger source.
148	:UNIT[1]|2|3|4:POWer	Query or set the unit of measured power
149	:UNIT[1]|2|3|4:POWer:RATio	Query or set the unit of measured ratio measurement power