Spectrum Analyzer

2398

Programming Manual
2398
Spectrum Analyzer
Programming Manual

Vol.2

Read this manual before using the equipment.
Keep this manual with the equipment.

IFR Americas, Inc.
Safety Symbols

Where these symbols or indications appear on the instrument or in this manual, they have the following meanings.

WARNING. Risk of hazard which may cause injury to human body or danger to life. If a WARNING appears on the instrument, and in this manual, do not proceed until its suitable conditions are understood and met.

CAUTION. Risk of hazard which caused fire or serious damage to the instrument or other equipment. Do not proceed until its suitable conditions are met.

GROUND. Ground terminal to chassis(earth).
For Safety

WARNING

1. ALWAYS refer to the operation manual when working near locations at which the alert mark shown on the left is attached. If the operation, etc., is performed without heeding the advice in the operation manual, there is a risk of personal injury. In addition, the equipment performance may be reduced. Moreover, this alert mark is sometimes used with other marks and descriptions indicating other dangers.

2. When supplying power to this equipment, connect the accessory 3-pin power cord to a 3-pin grounded power outlet. If a grounded 3-pin outlet is not available, use a conversion adapter and ground the green wire, or connect the frame ground on the rear panel of the equipment to ground. If power is supplied without grounding the equipment, there is a risk of receiving a severe or fatal electric shock and equipment damage.

3. This equipment cannot be repaired by the user. DO NOT attempt to open the cabinet or to disassemble internal parts. Only IFR trained service personnel or staff from your sales representative with a knowledge of electrical fire and shock hazards should service this equipment. There are high-voltage parts in this equipment presenting a risk of severe injury or fatal electric shock to untrained personnel. In addition, there is a risk of damage to precision parts.

4. This equipment should be used in the correct position. If the cabinet is turned on its side, etc., it will be unstable and may be damaged if it falls over as a result of receiving a slight mechanical shock.

For Safety

WARNING

Repair

WARNING

Falling Over
For Safety

CAUTION

1. Before changing the fuses, ALWAYS remove the power cord from the power outlet and replace the blown fuses. ALWAYS use new fuses of the type and rating specified on the fuse marking on the rear panel cabinet.

Changing Fuse

CAUTION

T3.15A indicates a time-lag fuse. There is risk of receiving a fatal electric shock if the fuses are replaced with the power cord connected.

2. Keep the power supply and cooling fan free of dust.
   - Clean the power inlet regularly. If dust accumulates around the power pins, there is a risk of fire.
   - Keep the cooling fan clean so that the ventilation holes are not obstructed. If the ventilation is obstructed, the cabinet may overheat and catch fire.

Cleaning

Check Terminal

3. Maximum DC voltage ratings:
   - RF Input 20Vdc
   - TG Output 0Vdc

   Maximum AC power ratings:
   - RF Input +26dBm
   - NEVER input a signal power >+26dBm or >± 20Vdc the to RF Input connector.
   - Excessive power may damage the internal circuits.
For Safety

CAUTION

4. The power for memory back-up is supplied by a Primary Lithium Battery. This battery should only be replaced by a battery of the same type (SANYO; CR12600SE-FT3); since replacement can only be made by IFR, contact the nearest IFR representative when replacement is required.

Note: The battery life is about 7 years. Early battery replacement is recommended.

Do not throw the battery away but dispose of it according to your country’s requirements.

CAUTION

5. This equipment stores data and programs using PCMCIA Type I Memory Card.

Data and programs may be lost due to improper use or failure. IFR therefore recommends that you back-up the memory.

IFR CANNOT COMPENSATE FOR ANY MEMORY LOSS.

Please pay careful attention to the following points.

- Do not remove the memory card from equipment being accessed.
- Isolate the card from static electricity.
- The back-up battery in the SRAM memory card has a limited life; replace the battery periodically.
For Safety

6. Use Proper Power Source: Do not operate this product from a power source that applies more than the voltage specified.

Provide Proper Ventilation: To prevent product overheating, provide proper ventilation.

Do Not Operate With Suspected Failures: If you suspect there is damage to this product, have it inspected by qualified service personnel.

Do Not Attempt To Operate If Protection May Be Impaired: If the instrument appears damaged or operates abnormally, protection may be impaired. Do not attempt to operate the instrument under these conditions. Refer all questions of proper instrument operation to qualified service personnel.

7. Object and Liquid Entry: Never push objects of any kind into instrument through openings as they may touch dangerous voltage points or short out parts that could result in a fire or electric shock. Never spill liquid of any kind on the instrument. Do not use this instrument near water— for example, near a bath tub, wash bowl, kitchen sink, or laundry tub, in a wet basement, or near a swimming pool, and the like. Keep the instrument away from damp air, water and dust. Unexpected trouble may be caused when the instrument is placed in a damp or dusty place.

Flammable and Explosive Substance: Avoid using this instrument where there are gases, and where there are flammable and explosive substances in the immediate vicinity.

Unstable Location: Do not place this instrument on an unstable cart, stand, tripod, bracket, or table. The instrument may fall, causing serious injury to a person, and serous damage to the instrument. Do not place or use the instrument in a place subject to vibration.
IFR Warranty

IFR will repair this equipment free of charge if a malfunction occurs within 1 year after shipment due to a manufacturing fault, provided that this warranty is rendered void under any or all of the following conditions.

- The fault is outside the scope of the warranty conditions described in the operation manual.
- The fault is due to misoperation, misuse, or unauthorized modification or repair of the equipment the customer.
- The fault is due to severe usage clearly exceeding normal usage.
- The fault is due to improper or insufficient maintenance by the customer.
- The fault is due to natural disaster including fire, flooding and earthquake, etc.
- The fault is due to use of non specified peripheral equipment, peripheral parts, consumables, etc.
- The fault is due to use of non specified power supply or a non-specified installation location.

In addition, this warranty is valid only for the original equipment purchaser. It is not transferable if the equipment is resold.

IFR will not accept liability for equipment faults due to unforeseen and unusual circumstances, nor for faults due to mishandling by the customer.

IFR Contact

If this equipment develops a fault, contact office of IFR at the address in the operation manual, or your nearest sales or service office.
Front Panel Power Switch

If the equipment is in the standby state, the front power switch of this equipment turns on the power when it is pressed.

If the switch is pressed continuously for one second in the power on state, the equipment enters the standby state to prevent malfunction caused by accidental touching.

In the power on state, if the power plug is removed from the outlet, then reinserted, the power will not be turned on. Also, if the line is disconnected due to momentary power supply interruption or power failure, the power will not be turned on even when power is restored.

This is because this equipment enters the standby state and prevents incorrect data from being acquired when the line has to be disconnected and reconnected.

For example, if the sweep time is 1,000 seconds and data acquisition requires a long time, momentary power supply interruption (power failure) might occur during measurement and the line could be recovered automatically to power on. In such a case, the equipment may mistake incorrect data for correct data without recognizing the momentary power supply interruption.

If this equipment enters the standby state due to momentary power supply interruption or power failure, check the state of the measuring system and press the front power switch to restore power to this equipment.

Further, if this equipment is built into a system and the system power has to be disconnected then reconnected, the power for this equipment must also be restored by pressing the front power switch.
DETECTION MODE

This instrument is a spectrum analyzer which uses a digital storage system. The spectrum analyzer makes level measurements in frequency steps obtained by dividing the frequency span by the number of measurement data points (500). Because of this operation it is desired to use the following detector modes associated with the appropriate measurements.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Detector mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal signal</td>
<td>POS PEAK</td>
</tr>
<tr>
<td>Random noise</td>
<td>SAMPLE OR AVERAGE</td>
</tr>
<tr>
<td>Pulsed noise</td>
<td>NORMAL</td>
</tr>
<tr>
<td>Occupied frequency bandwidth (for analog</td>
<td>SAMPLE</td>
</tr>
<tr>
<td>communication systems)</td>
<td></td>
</tr>
<tr>
<td>Occupied frequency bandwidth (for digital</td>
<td>POS PEAK or SAMPLE</td>
</tr>
<tr>
<td>communication systems)</td>
<td></td>
</tr>
</tbody>
</table>

When a detection mode is specified as one of the measurement methods, make the measurement in the specified detection mode.
ABOUT THIS MANUAL

(1) Composition of 2398 Manuals

The 2398 Spectrum Analyzer manuals of the standard type are composed of the following two documents. Use them properly according to the usage purpose.

- **Operation Manual**: Provides information on the 2398 outline, preparation before use, panel description, operation procedure, soft-key menu and performance tests.

- **Programming Manual**: The Remote Control Part provide information on RS-232C remote control, GPIB remote control and sample programs.
**TABLE OF CONTENTS**

SECTION 1 GENERAL ................................................................. 1- 1
  General description .................................................. 1- 3
  Remote control funtions ......................................... 1- 3
  Interface port selection functions .......................... 1- 3
  Examples of configurations using RS- 232C and GPIB ...... 1- 4
  Specifications of RS- 232C ....................................... 1- 5
  Specifications of GPIB ............................................. 1- 6

SECTION 2 CONNECTING DEVICE ........................................... 2- 1
  Connecting an external device with an RS- 232C cable ...... 2- 3
  Connection diagram of RS- 232C interface signals ......... 2- 4
  RS- 232C echo mode ............................................... 2- 4
  Setting the connection port interface ..................... 2- 5
  Setting the RS- 232C Interface conditions .................. 2- 6
  Connecting a device with a GPIB cable & requirements .. 2- 7
  Setting the GPIB address ....................................... 2- 8

SECTION 3 DEVICE MESSAGE FORMAT ..................................... 3- 1
  General description ............................................... 3- 3
  Program message format ...................................... 3- 3
  Response message format .................................. 3- 7

SECTION 4 DETAILED DESCRIPTION OF COMMANDS ....................... 4- 1
  General description ............................................... 4- 3
  Frequency .......................................................... 4- 4
  Span ................................................................. 4- 8
  Amplitude .................................................................. 4- 11
  Marker ....................................................................... 4- 15
  Trigger ....................................................................... 4- 23
  Coupling .................................................................... 4- 26
  Display control ....................................................... 4- 29
  Trace function .......................................................... 4- 33
    Mathematic ....................................................... 4- 38
    Detect ............................................................. 4- 41
    Average ............................................................ 4- 42
  Autoset ............................................................... 4- 44
  Save ........................................................................ 4- 45
  Recall ....................................................................... 4- 50
  Limit ......................................................................... 4- 53
  Window ...................................................................... 4- 56
  Measurement : dB DOWN ........................................... 4- 59
    Occupied power BandWidth measurement ................ 4- 63


Channel Power .................................................. 4- 64
Adjacent Channel Power(ACP) ............................... 4- 66
Quasi Peak mode(option) ..................................... 4- 70
Frequency counter ............................................. 4- 71
Auxiliary .......................................................... 4- 72
Preset ............................................................. 4- 74
Configuration : Printer ........................................ 4- 79
Clock set .......................................................... 4- 80
PCMCIA(option) ................................................... 4- 81
Reference clock ................................................... 4- 82
GPIB common command ......................................... 4- 83
Others ............................................................... 4- 88
Tracking Generator(option) ..................................... 4- 90

SECTION 5 STATUS STRUCTURE ........................................... 5- 1
IEEE488.2 standard status model ............................... 5- 3
Status byte(STB) register ....................................... 5- 5
ESB and MAV summary messages ............................... 5- 5
Device- dependent summary messages ......................... 5- 6
Reading and clearing the STB register ....................... 5- 7
Service request(SRQ) enabling operation ...................... 5- 8
Standard event status register ................................ 5- 9
Bit definition of standard event status register ............ 5- 9
Reading, writing, and clearing the standard event status register  .... 5- 10
Reading, writing, and clearing the standard event status enable register  .... 5- 10
Extended event status register ................................ 5- 11
Bit definition of END event status register ................ 5- 12
Reading, writing, and clearing the extended event status register .... 5- 13
Reading, writing, and clearing the extended event status enable register .... 5- 13

SECTION 6 EXAMPLE CODES ................................................. 6- 1
Frequency and Level at the marker point ..................... 6- 3
Delta Marker Measurement ..................................... 6- 5
Frequency Bandwidth ............................................ 6- 7
Occupied Bandwidth Measurement ............................. 6- 9
Marker Noise Measurement ..................................... 6- 11
Saving Data ....................................................... 6- 13
Recalling Data .................................................... 6- 14
Get Trace Data .................................................... 6- 15
Pass / Fail Check .................................................. 6- 17

APPENDIX

PROGRAMMING COMMANDS
ERROR CODE
SECTION 1
GENERAL

This section outlines the remote control and gives examples.

TABLE OF CONTENTS

General description .................................................................................................................. 1- 3
Remote control functions ........................................................................................................ 1- 3
Interface port selection functions ........................................................................................... 1- 3
Examples of configurations using RS- 232C and GPIB ......................................................... 1- 4
Specification of RS- 232C ...................................................................................................... 1- 5
Specification of GPIB .............................................................................................................. 1- 6
SECTION 1 GENERAL

General Description

The 2398 Spectrum Analyzer, when combined with an external controller (host computer, personal computer, etc.), can automate your measurement system. For this purpose, the 2398 is equipped with an RS- 232C interface port, GPIB interface bus.

Remote control functions

The remote control functions of the 2398 are used to do the following:
(1) Control all functions except the power switch and [LOCAL] key.
(2) Read setting value.
(3) Configure the automatic measurement system when the SA- 7270 is combined with a personal computer and other measuring instruments.
* Set the RS- 232C interface settings from the front panel.
* Set the GPIB address from the front panel.

Interface port selection functions

The 2398 Spectrum Analyzer has a standard RS- 232C interface, a GPIB interface bus and parallel(Printer) interface. Use the panel to select the interface port to be used to connect external devices as shown below.

Port for the external controller: Select RS- 232C or GPIB.
Port for the printer: Select parallel port.
Each interface can connect only one device.
Examples of configurations using RS-232C and GPIB

(1) Stand-alone type
Waveforms measured with the 2398 is output to the printer.

(2) Control by the host computer
The 2398 is controlled automatically or remotely from the computer.

(3) Control by the host computer
The waveforms measured by controlling 2398 automatically or remotely are output to the printer. The printer must be connected using printer port.
Specifications of RS-232C

The table below lists the standard specifications of RS-232C in the 2398.

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Control from the external controller (except for power- ON/OFF, [LOCAL] key)</td>
</tr>
<tr>
<td>Communication system</td>
<td>Asynchronous (start- stop synchronous system), half- duplex</td>
</tr>
<tr>
<td>Communication control system</td>
<td>none</td>
</tr>
<tr>
<td>Baud rate</td>
<td>600, 1200, 2400, 4800, 9600, 19200</td>
</tr>
<tr>
<td>Data bits</td>
<td>7 or 8 bits</td>
</tr>
<tr>
<td>Parity</td>
<td>Odd number (ODD), even number (EVEN), none (NON)</td>
</tr>
<tr>
<td>Start bit</td>
<td>1bit</td>
</tr>
<tr>
<td>Stop bit (bits)</td>
<td>1 or 2bits</td>
</tr>
<tr>
<td>Connector</td>
<td>D- sub 9- pin, female</td>
</tr>
</tbody>
</table>
Specifications of GPIB

The table below lists the specifications of the GPIB.

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification and supplementary explanation</th>
</tr>
</thead>
</table>
| **Function**          | Conforms to IEEE488.2  
The 2398 is controlled from the external controller (except for power- on/off, [LOCAL] key). |
| **Interface function**| SH1 : All source handshake functions are provided. Synchronizes the Timing of data transmission.           |
|                       | AH1 : All acceptor handshake functions are provided. Synchronizes the timing of data reception.             |
|                       | T6  : The basic talk functions and serial poll functions are provided. The talk only functions is not provided. The talker can be canceled by MLA. |
|                       | L4  : The basic listener functions are provided. The listen only function is not provided. The listener can be canceled by MTA. |
|                       | SR1 : All service request and status byte functions are provided.                                          |
|                       | RL1 : All remote/local functions are provided. The local lockout function is provided.                    |
|                       | PP0 : The parallel poll functions are not provided.                                                       |
|                       | DC1 : All device clear functions are provided.                                                             |
|                       | E2  : Output is tri- state.                                                                                |
|                       | LE0 : No extended listener capabilities                                                                   |
|                       | TE0 : No extended talker capabilities                                                                     |
SECTION 2
CONNECTING DEVICE

This section describes how to connect external devices such as the host computer, personal computer, with RS-232C, GPIB cables. This section also describes how to setup the interface of the 2398.

TABLE OF CONTENTS

Connection of an external device with an RS-232C cable .......................... 2-3
Connection diagram of RS-232C interface signals ........................................ 2-4
RS-232C echo mode .......................................................................................... 2-4
Setting the connection port interfaces .............................................................. 2-5
Setting the RS-232C interface conditions ......................................................... 2-6
Connecting a device with a GPIB cable & requirements ................................. 2-7
Setting the GPIB address .................................................................................. 2-8
SECTION 2 CONNECTING DEVICES

Connecting an external device with an RS-232C cable

Connect the RS-232C connector (D-sub 9-pin, male) on the rear panel of the 2398 to the RS-232C connector of the external device with an RS-232C cable.

Notes: RS-232C connectors with 9 pins are available. When purchasing the RS-232C cable, check the pins on the RS-232C connector of the external device. Also, the following RS-232C cables are provided as peripheral parts of the 2398.

- RS-232C cable (for personal computer)

<table>
<thead>
<tr>
<th>2398 side</th>
<th>Personal computer side</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-sub 9-pin, female</td>
<td>Length: 2m</td>
</tr>
<tr>
<td>(Cross)</td>
<td>D-sub 9-pin, female</td>
</tr>
</tbody>
</table>

---
SECTION 2 CONNECTING DEVICE

Connection diagram of RS-232C interface signals

The diagram below shows the RS-232C interface signal connections between 2398 and devices such as a personal computer.

<table>
<thead>
<tr>
<th>2398</th>
<th>Personal computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>CD(NC) 1</td>
<td>1 CD</td>
</tr>
<tr>
<td>RD 2</td>
<td>2 RD</td>
</tr>
<tr>
<td>TD 3</td>
<td>3 TD</td>
</tr>
<tr>
<td>DTR(NC) 4</td>
<td>4 DTR</td>
</tr>
<tr>
<td>GND 5</td>
<td>5 GND</td>
</tr>
<tr>
<td>DSR(NC) 6</td>
<td>6 DSR</td>
</tr>
<tr>
<td>RTS 7</td>
<td>7 RTS</td>
</tr>
<tr>
<td>CTS 8</td>
<td>8 CTS</td>
</tr>
<tr>
<td>RI(NC) 9</td>
<td>9 RI</td>
</tr>
</tbody>
</table>

< Connection with personal computer >

RS-232C echo mode

The serial port of 2398 may echo the received characters. ECHO is useful if 2398 is attached to a terminal. Echoing can be turned on or off by sending ECHO command. But the host must not echo characters received from 2398.
Setting the connection port interfaces

Set the interfaces between connection ports of the 2398 and devices such as a personal computer.

- **HARD COPY**
  - Prints hard copy.

- **HARD COPY SET..**
  - Selects HP Laserjet printer or PCMCIA CARD.

- **RS-232C CONFIG..**
  - Selects the Baud rate (600 to 19200), data length (7 or 8 bits), stop bit (bit 1 or 2), and parity (none, even, or odd) of the RS-232C.

- **CLOCK SET..**
  - Selects the year, month, day, hour, minute, second.

- **GPIB [0~31]**
  - Use the step or knob key to enter the GPIB address of this equipment.

- **PCMCIA..**
  - PCMCIA check and format.

- **MORE..**
Setting the RS-232C interface conditions

Sets the RS-232C interface conditions of this equipment to those of the external device to be connected.

- **BAUD [19200]**: Press this key to select a baud rate of 600, 1200, 2400, 4800, 9600, or 19200 bits per second.
- **DATA LEN [8]**: Press this key to select the number of data bits (7 or 8 bits).
- **STOP BIT [1]**: Press this key to select the number of stop bits (1 or 2 bits).
- **PARITY [NONE]**: Press this key to select the parity (none, even, or odd).
- **RETURN**
SECTION 2 CONNECTING DEVICE

Connecting a device with a GPIB cable & requirements

Connect the GPIB connector on the rear panel of this equipment to the GPIB connector of an external device with a GPIB cable.

Note: Be sure to connect the GPIB cable before turning the equipment power on.

GPIB Constraints:

1. Number of Interconnected Devices: 15 maximum
2. Interconnection Path Maximum Cable Length: 20 meters maximum or 2 meters per device (whichever is less).
3. Message Transfer Scheme: Byte serial, bit parallel a synchronous data transfer using a 3-line handshake system.
4. Data Rate: Maximum of 1 megabyte per second over the specified distances with tri-state drivers. Actual data rate depends on the transfer rate of the slowest device connected to the bus.
5. Address Capability: Primary address: 31 talk, 31 listen. A maximum of 1 talk and 14 listeners can be connected to the interface at given time.
6. Multiple- controller capability: In system with more than one controller, only one controller can be active at any given time. The active controller can pass control to another controller, but only the system controller can assume unconditional control. Only one system controller is allowed.
Setting the GPIB address

Set the GPIB address of this equipment as follows.

Use the step or knob key to enter the GPIB address of this equipment.
The initial value is 7.
SECTION 3
DEVICE MESSAGE FORMAT

This section describes the format of the device messages transmitted on the bus between a controller(host computer) and device(2398) via the RS-232C or GPIB system.

TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>General description</td>
<td>3-3</td>
</tr>
<tr>
<td>Program message format</td>
<td>3-3</td>
</tr>
<tr>
<td>Response message format</td>
<td>3-7</td>
</tr>
</tbody>
</table>
SECTION 3
DEVICE MESSAGE FORMAT

General description

The device messages are data messages transmitted between the controller and devices, program messages transferred from the controller to this instrument (device), and response messages input from this instrument (device) to the controller. There are also two types of program commands and program queries in the program message. The program command is used to set this instrument's parameters and to instruct it to execute processing. The program query is used to query the values of parameters and measured results.

Program message format

To transfer a program message from the controller program to this instrument using the "Send" statement, the program message formats are defined as follows.

EXAMPLE: Send(CF 1GHZ;);

PROGRAM MESSAGE: When the program message is transmitted from the controller to this instrument, the specified terminator is attached to the end of the program message to terminate its transmission.

(1) PROGRAM MESSAGE TERMINATOR

Carriage Return(CR) is ignored and is not processed as a terminator.
(2) PROGRAM MESSAGE

Multiple program message units can be output sequentially by separation them with a semicolon.
Example: Send( CF 1GHz; SP 500MHz; “);

(3) PROGRAM MESSAGE UNIT

The program header of an IEEE488.2 common command always begins with an asterisk. For numeric program data, the (SP) between the header and data can be omitted. The program header of a program query always ends with a question mark.

(4) PROGRAM DATA

(5) CHARACTER PROGRAM DATA

Character program data is specific character string data consisting of the upper- case alphabetic characters from A to Z, numbers 0 to 9, #, *, ?,

Example: Send( ST AUTO;”); ...... Sets Sweep Time to AUTO.
(6) NUMERIC PROGRAM DATA
Numeric program data has two types of formats: integer format (NR1) and fixed-point format (NR2).

**Integer format (NR1)**

- Zeros can be inserted at the beginning → 005, +000045
- There must be no spaces between a + or - sign and a number → +5, +Δ5 (x)
- Spaces can be inserted after a number → +5ΔΔΔ
- The + sign is optional → +5.5
- Commas cannot be used to separate digits → 1,234,567 (x)

**Fixed-point format (NR2)**

- The numeric expression of the integer format applies to the integer part.
- There must be no spaces between number and the decimal point → +753ΔΔ.123(x)
- Spaces can be inserted after the digits in the fraction part → +753.123ΔΔΔ
- A number need not be placed before the decimal point → .05
- A + or - sign can be placed before the decimal point → +.05, -.05
- A number can end with a decimal point → 12.
(7) SUFFIX PROGRAM DATA (unit)

The table below lists the suffixes used for the 2398.

Table of 2398 Suffix Codes

<table>
<thead>
<tr>
<th>Classification</th>
<th>Unit</th>
<th>Suffix code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frequency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GHZ</td>
<td>GHZ</td>
<td></td>
</tr>
<tr>
<td>MHz</td>
<td>MHZ</td>
<td></td>
</tr>
<tr>
<td>kHz</td>
<td>KHZ</td>
<td></td>
</tr>
<tr>
<td>Hz</td>
<td>HZ</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>HZ</td>
<td></td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Second</td>
<td>SEC</td>
<td></td>
</tr>
<tr>
<td>m second</td>
<td>MS</td>
<td></td>
</tr>
<tr>
<td>μ second</td>
<td>US</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>MS</td>
<td></td>
</tr>
<tr>
<td><strong>Level (dB system)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dB</td>
<td>DB</td>
<td></td>
</tr>
<tr>
<td>dBm</td>
<td>DBM</td>
<td></td>
</tr>
<tr>
<td>dBuV</td>
<td>DBUV</td>
<td></td>
</tr>
<tr>
<td>dBmV</td>
<td>DBMV</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>Determined in conformance with the set scale unit</td>
<td></td>
</tr>
<tr>
<td><strong>Level (V system)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>mV</td>
<td>MV</td>
<td></td>
</tr>
<tr>
<td>uV</td>
<td>UV</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>Determined in conformance with the set scale unit</td>
<td></td>
</tr>
<tr>
<td><strong>Level (W system)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>mW</td>
<td>MW</td>
<td></td>
</tr>
<tr>
<td>uW</td>
<td>UW</td>
<td></td>
</tr>
<tr>
<td>nW</td>
<td>NW</td>
<td></td>
</tr>
<tr>
<td>pW</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>Determined in conformance with the set scale unit</td>
<td></td>
</tr>
</tbody>
</table>
Response message format

To transfer the response messages from this instrument to the controller using the "Receive" statement, the response message formats are defined as follows.

(1) RESPONSE MESSAGE TERMINATOR

(2) RESPONSE MESSAGE

When a query is sent by the "Send" statement with one or more program queries, the response message also consists of one or more response message units.

(3) Usual RESPONSE MESSAGE UNIT
(4) RESPONSE DATA

(5) CHARACTER RESPONSE DATA
Character response data is specific character string data consisting of the upper-case alphabetic characters from A to Z, lower-case alphabetic characters from a to z, 0 to 9, and [, ], dot(.), minus(-), comma(,).

(6) NUMERIC RESPONSE DATA

< Integer format (NR1) >

< Example >
123
- 123

● The number at the most-significant position is other than 0.
● A + sign need not be placed before a positive number.

< Fixed-point format (NR2) >

● The number at the most-significant position is other than 0.
● A + sign need not be placed before a positive number.
SECTION 4
DETAILED DESCRIPTION OF COMMANDS

This section describes the usable device and response messages in functional order.

<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>General description</td>
</tr>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>Span</td>
</tr>
<tr>
<td>Amplitude</td>
</tr>
<tr>
<td>Marker</td>
</tr>
<tr>
<td>Trig</td>
</tr>
<tr>
<td>Coupling</td>
</tr>
<tr>
<td>Display control</td>
</tr>
<tr>
<td>Trace function</td>
</tr>
<tr>
<td>Mathematic</td>
</tr>
<tr>
<td>Detect</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Autoset</td>
</tr>
<tr>
<td>Save</td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>Limit</td>
</tr>
<tr>
<td>Window</td>
</tr>
<tr>
<td>Measurement:</td>
</tr>
<tr>
<td>dB DOWN</td>
</tr>
<tr>
<td>Occupied powerBand/Width measurement</td>
</tr>
<tr>
<td>Channel Power</td>
</tr>
<tr>
<td>Adjacent Channel Power(ACP)</td>
</tr>
<tr>
<td>Quasi Peak mode(option)</td>
</tr>
<tr>
<td>Frequency counter</td>
</tr>
<tr>
<td>Auxiliary</td>
</tr>
<tr>
<td>Preset</td>
</tr>
<tr>
<td>Configuration:</td>
</tr>
<tr>
<td>Printer</td>
</tr>
<tr>
<td>Clock set</td>
</tr>
<tr>
<td>PCMCIA(option)</td>
</tr>
<tr>
<td>Reference clock</td>
</tr>
<tr>
<td>GPIB common command</td>
</tr>
<tr>
<td>Others</td>
</tr>
<tr>
<td>Tracking Generator(option)</td>
</tr>
</tbody>
</table>
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

<BLANK>
SECTION 4
DETAILED DESCRIPTION OF COMMANDS

General Description

This section gives detailed descriptions of the device messages for the 2398 in functional order.

**Message headline**  | **Message name spelled**
--- | ---
CF | Center Frequency

- **Function**: Sets the center frequency and sets the spectrum analyzer to center frequency/span mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>CFΔf</td>
<td>CF?</td>
<td>CFΔf</td>
</tr>
<tr>
<td></td>
<td>CFΔa</td>
<td></td>
<td>f = 0 to 2.7GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transfers the data with suffix code.</td>
</tr>
</tbody>
</table>

- **Value of f**: 0Hz to 2.7GHz
- **Value of a**: OA, UP, DN
- **Suffix code**
  - None : Hz(10^0)
  - HZ : Hz(10^0)
  - KHZ : kHz(10^3)
  - MHZ : MHz(10^6)
  - GHZ : GHz(10^9)
- **Initial setting**: Value of f = 1.35GHz
- **Example**
  - CFΔ 123456;
  - CFΔ 50MHZ;
  - CF?;

Reference: “QRTYP” command
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

FREQUENCY

**CF**

Center Frequency

- **Function**: Sets the center frequency and sets the spectrum analyzer to center frequency / span mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>CFΔ f</td>
<td>CF?</td>
<td>CFΔ f</td>
</tr>
<tr>
<td></td>
<td>CFΔ a</td>
<td></td>
<td>f = 0 to 2.7GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transfers the data with suffix code.</td>
</tr>
</tbody>
</table>

- **Value of f**: 0Hz to 2.7GHz
- **Value of a**: OA : Function Query ( same as ? )
  UP : Increment size is 1/10 of the current span.
  DN : Decrement size is 1/10 of the current span.
- **Suffix code**: None : Hz(10^0)
  HZ : Hz(10^0)
  KHZ : kHz(10^3)
  MHZ : MHz(10^6)
  GHZ : GHz(10^9)
- **Initial setting**: Value of f = 1.35GHz
- **Example**: CF 123456;
  CF 50MHZ;
  CF?;
**FA**

**FA Start Frequency**

- **Function:** Sets the start frequency and sets the spectrum analyzer to start frequency / stop frequency mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA</td>
<td>FAΔf</td>
<td>FA?</td>
<td>FAΔf f = 0 to 2.7GHz Transfers the data with suffix code.</td>
</tr>
<tr>
<td></td>
<td>FAΔa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Value of f:** 0Hz to 2.7GHz
- **Value of a:**
  - OA: Function Query (same as ?)
  - UP: Increment size is 1/10 of the current span.
  - DN: Decrement size is 1/10 of the current span.
- **Suffix code:**
  - None: Hz(10^0)
  - HZ: Hz(10^0)
  - KHZ: kHz(10^3)
  - MHZ: MHz(10^6)
  - GHZ: GHz(10^9)
- **Initial setting:** Value of f = 0 GHz
- **Example:**
  - FA 123456;
  - FA 50MHZ;
  - FA?;
FB
FB
■ Function
Sets the stop frequency and sets the spectrum analyzer to start frequency / stop frequency mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>FB(\Delta f)</td>
<td>FB?</td>
<td>FB(\Delta f) f = 0 to 2.7GHz Transfers the data with suffix code.</td>
</tr>
<tr>
<td></td>
<td>FB(\Delta a)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

■ Value of \( f \) 0Hz to 2.7GHz
■ Value of \( a \) OA : Function Query ( same as ? )
UP : Increment size is 1/10 of the current span.
DN : Decrement size is 1/10 of the current span.
■ Suffix code None : Hz(10\(^0\))
HZ : Hz(10\(^0\))
KHZ : kHz(10\(^3\))
MHZ : MHz(10\(^6\))
GHZ : GHz(10\(^9\))
■ Initial setting Value of \( f = 2.7 \) GHz
■ Example FB 123456;
FB 50MHZ;
FB?;
SS

**SS**

Center Frequency Step Size

- **Function**: Sets the center frequency step size.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>SSΔ f</td>
<td>SS?</td>
<td>SSΔ f(AUTO), SSΔ f(MAN) f = 0 to 2.7GHz Transfers the data with suffix code.</td>
</tr>
<tr>
<td></td>
<td>SSΔ a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Value of f**: 0Hz to 2.7GHz
- **Value of a**
  - OA : Function Query (same as ?)
  - UP : Increment size is 1/10 of the current span.
  - DN : Decrement size is 1/10 of the current span.
  - AUTO : 10% of span
  - MAN : Manual

- **Suffix code**
  - None : Hz(10^0)
  - HZ : Hz(10^0)
  - KHZ : kHz(10^3)
  - MHZ : MHz(10^6)
  - GHZ : GHz(10^9)

- **Initial setting**: Value of f = 10% of span
- **Example**
  - SS 123456;
  - SS 50MHZ;
  - SS?;
## SPAN

**SP**

**Function**

Sets the frequency span.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>SPΔf</td>
<td>SP?</td>
<td>SPΔf</td>
</tr>
<tr>
<td></td>
<td>SPΔa</td>
<td></td>
<td>f = 0 to 2.7GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transfers the data with suffix code.</td>
</tr>
</tbody>
</table>

- **Value of f**: 0Hz to 2.7GHz
- **Value of a**:
  - OA: Function Query (same as ?)
  - UP: Increment the Parameter. 1, 2, 5, 10 sequence
  - DN: Decrement the Parameter. 1, 2, 5, 10 sequence
  - FULL: 2.7GHz
  - ZERO: 0MHz
  - LAST: Last Span
  - ZIN: Previous span / 2
  - ZOUT: Previous span x 2
- **Suffix code**:
  - None: Hz(10^0)
  - HZ: Hz(10^0)
  - KHz: kHz(10^-3)
  - MHZ: MHz(10^-6)
  - GHZ: GHz(10^-9)
- **Initial setting**: Value of f = 2.7GHz
- **Example**:
  - SP 123456;
  - SP 50MHZ;
  - SP?;
### FS

**FS**

**Function** Full Span

Selects the full frequency span.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>FS</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

FS;

---

### ZS

**ZS**

**Function** Zero Span

Sets zero frequency span.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZS</td>
<td>ZS</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

ZS;
### ZI

ZI  Zoom-in

**Function**  Changes to 1/2 the previous span.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZI</td>
<td>ZI</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**  ZI;

### ZO

ZO  Zoom-out

**Function**  Changes to two times the previous span.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZO</td>
<td>ZO</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**  ZO;
AMPLITUDE

**RL**

**RL** Reference Level
- **Function** Sets the reference level.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL</td>
<td>RL(\Delta) l</td>
<td>RL?</td>
<td>RL(\Delta) l</td>
</tr>
<tr>
<td></td>
<td>RL(\Delta) a</td>
<td></td>
<td>l; units value depending</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>on the current scale units.</td>
</tr>
</tbody>
</table>

- **Value of l** Value from -100dBm to +30dBm (0.1 step)
- **Value of a**
  - OA : Function Query (same as ?)
  - UP : Increment the Parameter. 1 division level
  - DN : Decrement the Parameter. 1 division level
- **Suffix code**
  - None : dBm
  - DBM : dBm
  - DBMV : dBmV
  - DBUV : dBuV
  - V : V
  - MV : mV
  - UV : uV
  - W : W
  - MW : mW
  - UW : uW
  - NW : nW
  - PW : pW
  - FW : fW

- **Initial setting** l = -10 dBm
- **Example**
  - RL 30 DBM;
  - RL UP;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

**AT**

**AT**

**Input Attenuation**

- **Function**: Sets the amount of attenuation for the input attenuator.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>AT(\Delta a)</td>
<td>AT?</td>
<td>AT(\Delta ) (\mathrm{dB})(AUTO)/ AT(\Delta ) (\mathrm{dB})(MAN)</td>
</tr>
<tr>
<td></td>
<td>AT(\Delta n)</td>
<td></td>
<td>n = 0, 10, 20, 30, 40, 50</td>
</tr>
</tbody>
</table>

- **Value of \(a\)**
  - AUTO : Auto
  - MAN : Manual
  - UP : Increment the Parameter. 10dB step.
  - DN : Decrement the Parameter. 10dB step.

- **Value of \(n\)**
  - 0 to 50 (6 step) : 0 to 50dB (10dB step)

- **Suffix code**
  - None : dB
  - DB : dB

- **Initial setting**
  - AT = Calculated value when AUTO is selected for AT

- **Example**
  - AT 10;

---

**LG**

**LG**

**Logarithm Amplitude Mode & Scale**

- **Function**: Selects 1, 2, 5, or 10 dB logarithmic amplitude mode & scale. When not in LOG mode, querying 'LG?' returns a zero.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG</td>
<td>LG(\Delta l)</td>
<td>LG?</td>
<td>LG(\Delta l) (\quad l = 0,1,2,5,10)</td>
</tr>
</tbody>
</table>

- **Value of \(l\)**
  - 1, 2, 5, 10 : dB/div

- **Suffix code**
  - None : 10 dB/div
  - DB : dB/div

- **Initial setting**
  - 10 dB/div

- **Example**
  - LG 10DB;
### LN

**LN**

**Function**

Selects linear amplitude mode. When not in linear mode, querying LN? returns a zero.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN</td>
<td>LN</td>
<td>LN?</td>
<td>LN(\Delta) sw</td>
</tr>
</tbody>
</table>

\[ \text{sw} = 0, 1 \]

- 1 : Linear Mode
- 0 : Not Linear Mode

**Value of sw**

- 0 : Not Linear Mode
- 1 : Linear Mode

**Suffix code**

None

**Example**

LN ;

---

### AUNITS

**AUNITS**

**Function**

Sets the absolute amplitude units for the input signal display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUNITS</td>
<td>AUNITS(\Delta) a</td>
<td>AUNITS?</td>
<td>AUNITS(\Delta) u</td>
</tr>
</tbody>
</table>

- \(u = \text{DBM}, \text{DBMV}, \text{DBUV}, \text{VOLT}, \text{WATT}\)

**Value of a**

- DBM : dBm
- DBMV : dBmV
- DBUV : dBuV
- V : Volt
- W : Watt

**Suffix code**

None

**Initial setting**

DBM: dBm

**Example**

AUNITS DBM;
## RLO

**RLO Reference Level Offset**

- **Function**: Set Reference Level Offset

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLO</td>
<td>RLO(\uparrow) l (=) RLO(\downarrow) a</td>
<td>RLO?</td>
<td>RLO(\uparrow) res (= xx.x dB)</td>
</tr>
</tbody>
</table>

- **Value of \(l\)**: -10.0 dB ~ +10.0 dB
- **Value of \(a\)**:
  - OA: Function Query (same as ?)
  - UP: Increment size: 1 dB
  - DN: Decrement size: 1 dB
- **Suffix code**: None: dB
- **Initial setting**: \(l = 0.0 \text{ dB}\)
- **Example**: RLO 6 dB;
MARKER

**MKN**

<table>
<thead>
<tr>
<th>MKN</th>
<th>Normal Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Places an active marker on the specified frequency. If no frequency is specified, MKN places the marker at the center of trace. When zero span, the marker is set at the specified time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKN</td>
<td>MKNΔf</td>
<td>MKN?</td>
<td>MKNΔf f = 0 to 2.7GHz</td>
</tr>
<tr>
<td></td>
<td>MKNΔa</td>
<td></td>
<td>MKNΔt t = 0 to 20SEC (time resolution: sweep time/500)</td>
</tr>
<tr>
<td></td>
<td>MKNΔt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Value of **f**: 0Hz to 2.7GHz
- Value of **t**: 0 to 20 sec
- Value of **a**: OA: Function Query (same as?)
  UP: Increment the Parameter. 10% of span
  DN: Decrement the Parameter. 10% of span
  None: When normal marker is not specified, put the normal marker on the center on grid.

- Suffix code **f**:
  None: Hz(10^0), In sweep mode
  HZ: Hz(10^0)
  KHZ: kHz(10^3)
  MHZ: MHz(10^6)
  GHZ: GHz(10^9)

- **t**:
  None: ms(10^-3), In zero mode
  US: us(10^-6),
  MS: ms(10^-3)
  SEC: sec(10^0)

- Initial setting: OFF
- Example: MKN;
  MKN 100MHZ;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

**MKA**

**MKA Marker Amplitude**

**Function**

Returns on the amplitude data in marker mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKA</td>
<td>- - - - -</td>
<td>MKA?</td>
<td>MKA(\Delta I)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MKA(\Delta v)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MKA(\Delta w)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MKA(\Delta f)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MKA(\Delta p)</td>
</tr>
</tbody>
</table>

- **Value of I** when display unit system for marker level is dB.
- **Value of v** when display unit system for marker level is V.
- **Value of w** when display unit system for marker level is W.
- **Value of f** For FM Demodulation, kHz
- **Value of p** For AM Demodulation, %
- **Example** MKA?;

---

**MKDTF**

**MKD Marker 1/ Delta**

**Function**

Calculates 1/ delta in the zero span mode or sweep mode

The normal & delta marker must be on to work.

The only way to turn MKDTF off is to turn the marker off (MKOFF).

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKDTF</td>
<td>MKDTF</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

- **Example** MKDTF;
### MKD

**MKD**  
Marker Delta  
**Function**  
Places delta marker on the normal marker position.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKD</td>
<td>MKD</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

**Example**  
MKD;

### MKTF

**MKTF**  
Read the marker frequency or time  
**Function**  
Returns time or frequency of a marker.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
</table>
| MKTF   | - - -           | MKTF? | MKTFΔ f(frequency)  
MKTFΔ t(time)  
Transfers the data with suffix code. |

**Example**  
MKTF?;

### MKOFF

**MKOFF**  
Marker Off  
**Function**  
Turns off the marker mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKOFF</td>
<td>MKOFF</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

**Example**  
MKOFF;
### MKPK

**MKPK**

**Peak Search**

**Function** Places a marker on the highest point on the trace, the next-highest point, the next-left peak, the next-right peak.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKPK</td>
<td>MKPK(\Delta a)</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Value of a**

- **HI**: Finds the highest point on the trace.
- **NH**: Finds the next-highest point on the trace.
- **NR**: Finds the next-right peak.
- **NL**: Finds the next-left peak.
- **None**: Finds the highest point on the trace.

**Example**

- MKPK;
- MKPK HI;
- MKPK NH;

### MKCF

**MKCF**

**Marker to Center Frequency**

**Function** Sets the center frequency to the frequency value of the normal marker. The normal marker must be active to work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKCF</td>
<td>MKCF</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

- MKCF;
MKRL

**MKRL**

Marker to Reference Level

**Function**

Sets the reference level to the amplitude of the normal marker. The normal marker must be active to work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKRL</td>
<td>MKRL</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

MKRL;

---

MKSP

**MKSP**

Marker Delta to Span

**Function**

Sets the frequency span equal to the frequency difference between two markers on a trace. If normal & delta marker is not active, MKSP cannot work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKSP</td>
<td>MKSP</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

MKSP;
MKSS
MKSS Marker to Center Frequency Step-size
Function Sets the center frequency step-size equal to the frequency value of the active marker.
The normal marker must be active to work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKSS</td>
<td>MKSS</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

Example MKSS;

MKZI
MKZI Marker Zoom-in
Function Sets the center frequency to the frequency value of an active marker and the frequency span changes to 1/2 the previous span. The normal marker must be active to work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKZI</td>
<td>MKZI</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

Example MKZI;
MKZO
MKZO Marker Zoom-out

- Function
Sets the center frequency to the frequency value of an active marker and the frequency span changes to the two times the previous span. The normal marker must be active to work.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKZO</td>
<td>MKZO</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Example
MKZO:

MKTRACK
MKTRACK Signal Track

- Function
Locates the active marker and sets the center frequency to the marker value. This is done after sweep, thus maintaining the marker value at the center frequency.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKTRACK</td>
<td>MKTRACK(\text{(\text{\text{\text{\text{sw}}}}))}</td>
<td>MKTRACK?</td>
<td>MKTRACK(\text{(\text{(\text{\text{\text{sw}}}}))} \text{(\text{\text{\text{\text{sw=0,1}}}}))}</td>
</tr>
</tbody>
</table>

- Value of sw
1, ON : On
0, OFF : Off

- Suffix code
None

- Initial setting
0, OFF

- Example
MKTRACK ON;
MKNOISE

MKNOISE Marker Noise

- **Function**: Sets the detector mode to sample and computes the value between the normal and the delta marker.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKNOISE</td>
<td>MKNOISE(\Delta) (sw)</td>
<td>MKNOISE?</td>
<td>MKNOISE(\Delta) (rsw) (rsw = OFF), Result Value</td>
</tr>
</tbody>
</table>

- **Value of \(sw\)**
  - ON : On
  - OFF : Off

- **Value of \(rsw\)**
  - OFF : Off
  - Result Value and Suffix code is dBC/Hz

- **Suffix code**: None
- **Initial setting**: OFF
- **Example**: MKNOISE ON;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

TRIG

TRGSWP

TRGSWP Trigger Sweep
- Function Selects the continuous-sweep mode or the single-sweep mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRGSWP</td>
<td>TRGSWPΔ sw</td>
<td>TRGSWP?</td>
<td>TRGSWPΔ sw  sw = 0, 1</td>
</tr>
</tbody>
</table>

- Value of sw
  - 0, CNT : Continuous- sweep Mode
  - 1, SNG : Single- sweep Mode
- Suffix code None
- Initial setting
  - 0, CNT : Continuous- sweep Mode
- Example TRGSWP 0;

TM

TM Trigger Source
- Function Sets the trigger switch and trigger source.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>TMΔ sw</td>
<td>TM?</td>
<td>TMΔ sw  sw = FREE, VID, LINE, EXT</td>
</tr>
</tbody>
</table>

- Value of sw
  - FREE : Selects the free-run mode.
  - VID : Selects the video mode.
  - LINE : Selects the line mode.
  - EXT : Selects the external mode.
- Suffix code None
- Initial setting FREE
- Example TM FREE;
  TM VID;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

**TF**

**TF**

**Trigger Filter**

- **Function**: Selects the trigger filter.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF</td>
<td>TF(\Delta) sw</td>
<td>TF?</td>
<td>TF(\Delta) sw  sw = HPF, LPF</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - HPF: Selects the high pass filter.
  - LPF: Selects the low pass filter.
- **Suffix code**: None
- **Initial setting**: LPF
- **Example**: TF HPF;
  - TF LPF;

---

**TLV**

**TLV**

**Trigger Level**

- **Function**: Sets the threshold level of sweep the start trig when the trigger source is video. Sweep trigger level \(x\) is vertical position on graticule and ranges from 0 to 255 (0 is Bottom).

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLV</td>
<td>TLV(\Delta) x</td>
<td>TLV?</td>
<td>TLV(\Delta) x  (x = 0) to 255</td>
</tr>
</tbody>
</table>

- **Value of x**: \(x\) : 0 to 255
- **Suffix code**: None
- **Initial setting**: 0
- **Example**: TLV 100;
  - TLV?;
**TDLY**

**TDLY**

**Delay Time**

**Function**

Sets the delay time from point where trace time triggering occurs. Available only zero span mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDLY</td>
<td>TDLYΔ t</td>
<td>TDLY?</td>
<td>TDLYΔ t</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>t = - Sweep Time &lt; t &lt; Sweep Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Resolution: Sweep time / 500</td>
</tr>
</tbody>
</table>

- **Value of t**
  
  t : - Sweep Time < t < Sweep Time

  Resolution: Sweep time / 500

- **Suffix code**
  
  None : ms

  US : us

  MS : ms

  SEC : sec

- **Initial setting**
  
  0 : 0 sec

- **Example**
  
  TDLY 50MS;
  
  TDLY?;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

COUPLING

**AUTOCPL**

AUTOCPL

- Function
  
  Sets the resolution bandwidth, the video bandwidth, the input attenuator, and the sweep time in AUTO mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOCPL</td>
<td>AUTOCPL</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

**Example**

AUTOCPL;

**RB**

RB

- Function
  
  Sets the resolution bandwidth.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB</td>
<td>RB(\Delta) a</td>
<td>RB?</td>
<td>RB(\Delta) f(AUTO), RB(\Delta) f(MAN)</td>
</tr>
<tr>
<td></td>
<td>RB(\Delta) f</td>
<td></td>
<td>(f = 300) to (3\text{MHZ})</td>
</tr>
</tbody>
</table>

- Value of a
  
  UP : Increments in a 1, 3, 10 sequence.
  
  DN : Decrements in a 1, 3, 10 sequence.
  
  OA : Function Query (same as ?)
  
  AUTO : RBW Auto coupling
  
  MAN : RBW Manual coupling

- Value of f
  
  300Hz to 3MHz (1, 3, 10 sequence.)

- Suffix code f
  
  None : Hz(10^0)
  
  HZ : Hz(10^0)
  
  KHZ : kHz(10^3)
  
  MHZ : MHz(10^6)

- Initial setting
  
  RBW = calculated value when AUTO is selected for RBW.

- Example
  
  RB 3KHZ;
### VB

**VB**

- **Video Bandwidth**
- **Function** Sets the video bandwidth.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>VBΔ a</td>
<td>VB?</td>
<td>VBΔ f (AUTO), VBΔ f (MAN)</td>
</tr>
<tr>
<td></td>
<td>VBΔ f</td>
<td></td>
<td>f = 10 to 1MHz, NONE</td>
</tr>
</tbody>
</table>

- **Value of a**
  - UP: Increments in a 1, 3, 10 sequence.
  - DN: Decrements in a 1, 3, 10 sequence.
  - OA: Function Query (same as ?)
  - AUTO: VBW Auto coupling
  - MAN: VBW manual coupling
  - NONE: Not Filtering

- **Value of f** 10Hz to 1MHz (1, 3, 10 sequence.)

- **Suffix code f:**
  - None: Hz(10^0)
  - HZ: Hz(10^0)
  - KHZ: kHz(10^-3)
  - MHZ: MHz(10^-6)

- **a:** NONE

- **Initial setting** VBW = calculated value when AUTO is selected for VBW.

- **Example** VB 3KHZ;
ST

Function: Sweep Time

ST \(\triangle a\)

Sets the sweep time.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
<td>(\triangle a)</td>
<td>ST?</td>
<td>(\triangle t) (AUTO), (\triangle t) (MAN)</td>
</tr>
<tr>
<td></td>
<td>(\triangle t)</td>
<td></td>
<td>(t=50) ms to 10000 (sweep)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(t=5) ms to 200 (zero span)</td>
</tr>
</tbody>
</table>

- Value of a
  - UP: Increments in a 1, 2, 5, 10 sequence.
  - DN: Decrements in a 1, 2, 5, 10 sequence.
  - OA: Function Query (same as ?)
  - AUTO: Sweep time Auto coupling
  - MAN: Sweep time Manual coupling

- Value of t: 50 ms to 1000 sec (5 ms to 20 sec for Zero Span.)

- Suffix code t:
  - None: ms(10^-3)
  - MS: ms(10^-3)
  - SEC: sec(10^0)

- Initial setting: ST = calculated value when AUTO is selected for Sweep time.

- Example
  - ST AUTO;
  - ST 20MS;
DISPLAY CONTROL

**DL**

**DL**

Display Line

**Function**

Activates a horizontal line for use as a visual aid or for computational purposes.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL</td>
<td>DL Δ sw</td>
<td>DL?</td>
<td>DL Δ rsw</td>
</tr>
<tr>
<td></td>
<td>DL Δ l</td>
<td></td>
<td>rsw = OFF, l (real)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - OFF : Off
  - ON : On
  - OA : Function Query (same as ?)

- **Value of l**
  - Number : real. Dependent upon the selected amplitude units.

- **Suffix code**
  - DBM : dBm
  - DBMV : dBmV
  - DBUV : dBuV
  - V : V (MV : mV, UV : uV)
  - KHZ : FM Mode
  - None : AM (%)

- **Initial setting**
  - OFF : Off

- **Default value**
  - bottom level.

- **Example**
  - DL - 50DBM;
**TH**

**TH**

Threshold

- **Function**
  Sets the minimum amplitude level and ignores data below this value.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH</td>
<td>TH(\Delta) sw</td>
<td>TH?</td>
<td>TH(\Delta) rsw</td>
</tr>
<tr>
<td></td>
<td>TH(\Delta) l</td>
<td></td>
<td>rsw = OFF, l(real)</td>
</tr>
</tbody>
</table>

- **Value of** \(\Delta\) **sw**
  - OFF: Off
  - ON: On
  - OA: Function Query (same as ?)

- **Value of** l
  Number: real. Dependent upon the selected amplitude units.

- **Suffix code**
  - DBM: dBm
  - DBMV: dBMV
  - DBUV: dBuV
  - V: V (MV, mV, UV, uV)
  - W: W (MW, mW, uW, nW, PW, pW, fW, fW)
  - KHZ: FM Mode
  - None: AM (%)

- **Initial setting**
  - OFF: Off

- **Default value**
  - bottom level.

- **Example**
  - TH - 50DBM;
## TITLE

### TITLE

**Screen Title Entry**

- **Function**: Places character data in the title area of the display.
  - Available characters are Alpha-numeric.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
</table>
| TITLE  | TITLE{text}     | TITLE?| TITLE{text};
  - text = ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - 0123456789 (8 characters) |

- **Value of text**: Character string within 8 characters.
- **Example**: TITLE IF2398;
  - TITLE SPECTRUM;

## CNTRST

### CNTRST

**Contrast Control**

- **Function**: Controls the display contrast.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTRST</td>
<td>CNTRST{l}</td>
<td>CNTRST?</td>
<td>CNTRST{l};</td>
</tr>
</tbody>
</table>
  - l = 0 to 145 |

- **Value of l**: Contrast level 0 to 145 (5 Step)
- **Suffix code**: None
- **Initial setting**: 50
- **Example**: CNTRST 50;
### GRAT

**GRAT**

Graticule On/Off

- **Function**: Turns the display graticule on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAT</td>
<td>GRAT△ sw</td>
<td>GRAT?</td>
<td>GRAT△ sw sw=0,1</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 0, OFF : Off
  - 1, ON : On

- **Initial setting**: ON

- **Example**: GRAT ON;

### ANNOT

**ANNOT**

Annotation On / Off

- **Function**: Turns the display annotation on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANNOT</td>
<td>ANNOT△ sw</td>
<td>ANNOT?</td>
<td>ANNOT△ sw sw=0,1</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 1, ON : On
  - 0, OFF : Off

- **Suffix code**: None

- **Initial setting**: ON

- **Example**: ANNOT ON;
TRACE FUNCTION

CLEW

CLEW Clear Write
■ Function Sets the chosen trace to clear-write mode.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEW</td>
<td>CLEW</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

■ Suffix code None
■ Example CLEW;

VIEW

VIEW View Trace
■ Function Displays the current contents of the selected trace memory, but does not update the memory contents.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW</td>
<td>VIEW</td>
<td>-</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

■ Suffix code None
■ Example VIEW;
### MXMH

**MXMH**

- **Function**: Maximum Hold
- Updates the chosen trace with the maximum signal level detected at each trace-data point from subsequent sweeps.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MXMH</td>
<td>MXMH</td>
<td>-</td>
<td>- - - - -</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: MXMH;

---

### BLANK

**BLANK**

- **Function**: Blank Trace
- Blanks the trace from the display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLANK</td>
<td>BLANK</td>
<td>-</td>
<td>- - -</td>
</tr>
</tbody>
</table>

- **Example**: BLANK;
**TRA / TRB**

**Function**
Provides a method for transferring trace data to or from a computer. The available data formats are decimal number(d) format, binary(b) format (only GPIB).

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>TRA#ns#ne,&lt;b0...bn&gt;</td>
<td>TRA#ns#ne,?</td>
<td>b = Binary data (2 bytes)</td>
</tr>
<tr>
<td>TRB</td>
<td>TRB#ns#ne,&lt;d0...dn&gt;</td>
<td>TRB#ns#ne,?</td>
<td>[b1b2...bn]</td>
</tr>
<tr>
<td></td>
<td>TRA#ns#ne,&lt;b0...bn&gt;</td>
<td></td>
<td>d = Decimal data (ASCII code)</td>
</tr>
<tr>
<td></td>
<td>TRB#ns#ne,&lt;d0...dn&gt;</td>
<td></td>
<td>[d1,d2,...dn]</td>
</tr>
</tbody>
</table>

- **Value of ns, ne** 1 to 500 (Point NO.)
- **Value of b**
  - LOG : 0000h to 0E10h
  - Linear : 0000h to 0708h
  - AM, FM : 0000h to 0E10h
  - QP MODE : 0000h to 0FA0h
- **Value of d**
  - LOG : 0 to 3600
  - Linear : 0 to 1800
  - AM, FM : 0 to 3600
  - QP MODE : 0 to 4000
- **Suffix code** None
- **Initial setting** Decimal data format
- **Example**
  - TRA #1 #3, <2048, 1248, 200>
  - TRA #1 #3, <0ABC09870778>
  - TRA #1 #500,
## TRAALL / TRBALL

**TRAALL / TRBALL**  
Trace All Data Output

**Function**  
Provides a method for transferring all trace data to the computer. The available data formats are decimal number(d) format, binary(b) format (only GPIB).

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAALL</td>
<td>-</td>
<td>TRAALL?</td>
<td>b = Binary data (2 bytes)</td>
</tr>
<tr>
<td>TRBALL</td>
<td>-</td>
<td>TRBALL?</td>
<td>[b1b2...b1000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d = Decimal data (ASCII code)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[d1,d2,...d500]</td>
</tr>
</tbody>
</table>

- **Value of b**
  - LOG : 0000h to 0E10h  
  - Linear : 0000h to 0708h  
  - AM, FM : 0000h to 0E10h  
  - QP MODE : 0000h to 0FA0h
- **Value of d**
  - LOG : 0 to 3600  
  - Linear : 0 to 1800  
  - AM, FM : 0 to 3600  
  - QP MODE : 0 to 4000
- **Suffix code**
  - None
- **Initial setting**
  - Decimal data format
- **Example**
  - TRAALL?;  
  - TRBALL?;
TDF

TDF Trace Data Format

Selects the format for input and output trace data.
You must specify the desired format when transferring data from the spectrum analyzer to a computer.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDF</td>
<td>TDFΔ sw</td>
<td>TDF?</td>
<td>TDFΔ sw  sw = BIN, DEC</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - BIN : Binary data format
  - DEC : Decimal data format (ASCII Code)

- **Suffix code**
  - None

- **Initial setting**
  - Decimal

- **Example**
  - TDF BIN;

TRSTS

TRSTS Trace Status

Sets the trace status.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRSTS</td>
<td>TRSTSΔ sw</td>
<td>TRSTS?</td>
<td>TRSTS A=rsw B=rsw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rsw = 0, 1, 2, 3</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - TRA : Trace A
  - TRB : Trace B

- **Value of rsw**
  - 0 : Clear write
  - 1 : View
  - 2 : Blank
  - 3 : Max. Hold

- **Suffix code**
  - None

- **Initial setting**
  - None

- **Example**
  - TRSTS TRA;
  - TRSTS ?;
MATHEMATIC

**AMB**

AMB

- **Function**
- Trace A Minus Trace B
  - Subtracts the contents of Trace B from Trace A and places the result in Trace A.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMB</td>
<td>AMB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**

AMB;

---

**BML**

BML

- **Function**
- Trace B Minus Display Line
  - Subtracts the display line from Trace B and places the result in Trace B.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>BML</td>
<td>BML</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**

BML;
**APB**

**APB**

<table>
<thead>
<tr>
<th>Function</th>
<th>Trace A Plus Trace B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds the contents of Trace B to Trace A and stores the result in Trace A.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB</td>
<td>APB</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

| Example | APB; |

**AMBPL**

**AMBPL**

<table>
<thead>
<tr>
<th>Function</th>
<th>Trace A Minus Trace B Plus Display Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtracts the contents of Trace B from Trace A, adds the display line to this value, and stores the result in Trace A.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMBPL</td>
<td>AMBPL</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

| Example | AMBPL; |
AXB

Trace A Exchange Trace B

- **Function**
  Exchanges the contents of Trace A with those of Trace B.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXB</td>
<td>AXB</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

- **Example**
  AXB;
## DETECT MODE

**DET**

**DET**

**Function** Detection Mode

Selects the detection mode for the waveform data being displayed.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
</table>
| DET    | DET<sub>_d</sub> | DET?  | DET<sub>_d</sub>
|        |                 |       | D = POS, SAM, NEG, NRM, AVG |

- **Value of d**
  - POS : Positive Peak
  - NEG : Negative Peak
  - SAM : Sample
  - NRM : Normal
  - AVG : Average

- **Suffix code** None
- **Initial setting** POS : Positive Peak
- **Example** DET NRM;

---

4-41
### AVERAGE

**AVG**

**Function**
- Trace Average

**AVG Trace Average**

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>AVG△ sw</td>
<td>AVG?</td>
<td>AVG△ sw  sw = 0, 1</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 1, ON : On
  - 0, OFF : Off

- **Suffix code**
  - None

- **Initial setting**
  - OFF

- **Example**
  - AVG ON;

### AVGC

**AVGC**

**Function**
- Sets the averaging rate (number of sweep repetitions).

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVGC</td>
<td>AVGC△ n</td>
<td>AVGC?</td>
<td>AVGC△ n  n = 2 to 256</td>
</tr>
</tbody>
</table>

- **Value of n**
  - 2 : 2 times
  - 3 : 3 times
  - ..
  - 256 : 256 times

- **Suffix code**
  - None

- **Initial setting**
  - 8 : 8 times

- **Example**
  - AVGC 32;
AVGCYL
AVGCYL  Average Cycle On or Off
Function  Set averaging cycle on means that the trace stops after the number of times of averaging has completed.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVGCYL</td>
<td>AVGCYLΔ sw</td>
<td>AVGCYL?</td>
<td>AVGCYLΔ sw sw=0,1</td>
</tr>
</tbody>
</table>

- Value of sw
  - 1, ON : On
  - 0, OFF : Off
- Suffix code: None
- Initial setting: OFF
- Example: AVGCYL ON;

AVGS
AVGS  Average Control
Function  Set averaging cycle to stop, continue or reset.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVGS</td>
<td>AVGSΔ sw</td>
<td>AVGS?</td>
<td>AVGSΔ sw sw = STOP, CONT</td>
</tr>
</tbody>
</table>

- Value of sw
  - STOP : Stop
  - CONT : Continue
  - RSET : Reset
- Suffix code: None
- Initial setting: CONT
- Example: AVGS CONT;
# AUTOSET

**AUTOSET**

Auto Sets

- **Function**: Detects the maximum peak point in full span, and displays its spectrum in the center of the screen, and then changes to a small span width, and the last span width to 1MHz.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSET</td>
<td>AUTOSET</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>
SAVE

**SVS**

**Function**
Save State into Internal Register

- Saves the current- displayed instrument state in the specified register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVS</td>
<td>SVS_n</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Value of n: 0 to 9 (Memory number)
- Suffix code: None
- Example: SVS 1;

**SVMS**

**Function**
Save State into Memory card

- Saves the current- displayed instrument state in the specified Memory card. File name is TITLE text and extension is automatically made as STS. When the same file is already in the memory card, it will OVERWRITE the file.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMS</td>
<td>SVMS</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Suffix code: None
- Example: SVMS;
SVTRA

SVTRA Save Trace A into Internal Register

- Function Saves Trace A in the specified trace register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVTRA</td>
<td>SVTRA(\Delta n)</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

- Value of \(n\) 0 to 9 (Memory number)
- Suffix code None
- Example SVTRA 1;

SVMTRA

SVMTRA Save Trace A into Memory Card

- Function Saves Trace A in the specified Memory card. File name is TITLE text and extension is automatically made as TRA. When the same file is already in the memory card, it will OVERWRITE the file.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMTRA</td>
<td>SVMTRA</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

- Suffix code None
- Example SVMTRA;
SVTRB

Save Trace B into Internal Register

**Function**

Saves Trace B in the specified trace register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVTRB</td>
<td>SVTRB n</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 9 (Memory number)
- **Suffix code**: None
- **Example**: SVTRB 1;

SVMTRB

Save Trace B into Memory Card

**Function**

Saves Trace B in the specified Memory card. File name is TITLE text and extension is automatically made as TRB. When the same file is already in the memory card, it will OVERWRITE the file.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMTRB</td>
<td>SVMTRB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: SVMTRB;
**SVLMT**

**SVLMT**  
Save Mask data into Internal Register  
**Function**  
Saves the current displayed mask data in the specified register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVLMT</td>
<td>SVLMT(\Delta n)</td>
<td>- - -</td>
<td>- - - - -</td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 9 (Memory number)
- **Suffix code**: None
- **Example**: SVLMT 1;

---

**SVMLMT**

**SVMLMT**  
Save Mask data into Memory Card  
**Function**  
Saves the current displayed mask data in the specified Memory card. File name is TITLE text and extension is automatically made as LMT. When the same file is already in the memory card, it will OVERWRITE the file.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMLMT</td>
<td>SVMLMT</td>
<td>- - -</td>
<td>- - - - -</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: SVMLMT;
SVLCK

SVLCK
Save Lock on or off

Function
Do or don’t save in internal register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVLCK</td>
<td>SVLCK(\Delta) sw</td>
<td>SVLCK?</td>
<td>SVLCK(\Delta) sw (sw = 0, 1)</td>
</tr>
</tbody>
</table>

- Value of \(sw\): 1, ON : On
  0, OFF : Off
- Suffix code: None
- Initial setting: OFF
- Example: SVLCK ON;
RECALL

RCS

RCS
Recall State from Internal Register
■ Function
Recalls to the display a previously- saved instrument state.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCS</td>
<td>RCSΔ n</td>
<td>- - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

■ Value of n      0 to 9 (Memory number)
■ Suffix code     None
■ Example         RCS 1;

RCTRA

RCTRA
Recall Trace A from Internal Register
■ Function
Recalls previously- saved trace data to the display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCTRA</td>
<td>RCTRAΔ n</td>
<td>- - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

■ Value of n      0 to 9 (Memory number)
■ Suffix code     None
■ Example         RCTRA 1;
**RCTRB**

**RCTRB**
Recall Trace B from Internal Register

- **Function**: Recalls previously saved trace data to the display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCTRBL</td>
<td>RCTRBL{n}</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 9 (Memory number)
- **Suffix code**: None
- **Example**: RCTRBL 1;

---

**RCLMT**

**RCLMT**
Recall Mask data from Internal Register

- **Function**: Recalls previously saved mask data to the display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCLMT</td>
<td>RCLMT{n}</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 9 (Memory number)
- **Suffix code**: None
- **Example**: RCLMT 1;
RCM

RCM Recall Data from Memory Card

- **Function**: Recalls the measurement conditions (parameters) and measured results (traces) from memory card.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCM</td>
<td>RCM t.e</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

- **Value of t**
  - **Text**: Character string within 8 characters

- **Value of e**
  - **Text**:
    - TRA: Trace A
    - TRB: Trace B
    - STS: State (Measurement condition)
    - LMT: Mask data (Limit line)

- **Suffix code**: None
- **Example**: RCM SPECTRUM.TRA;
**LIMIT**

**LMTPC**

LMTPC Limit Line Function On / Off

- **Function**
  Sets the limit line function on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMTPC</td>
<td>LMTPC(\triangle) sw</td>
<td>LMTPC?</td>
<td>LMTPC(\triangle) rs(w)</td>
</tr>
</tbody>
</table>

\(rs\(w\) = OFF, UFIL, LFIL, PASS, AFIL\)

- **Value of sw**
  - ON; On
  - OFF; Off

- **Value of rs\(w\)**
  - OFF; Off
  - UFIL; Upper Failure
  - LFIL; Lower Failure
  - PASS; Pass
  - AFIL; All Failure

- **Suffix code**
  None

- **Initial setting**
  OFF

- **Example**
  LMTPC ON;

**ALARM**

ALARM Pass/Fail Alarm

- **Function**
  Sets alarm when limit- pass/fail check

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALARM</td>
<td>ALARM(\triangle) sw</td>
<td>ALARM?</td>
<td>ALARM(\triangle) rs(w)</td>
</tr>
</tbody>
</table>

\(rs\(w\) = ON, OFF\)

- **Value of sw**
  - ON; On
  - OFF; Off
  - OA; Function Query (same as ?)

- **Suffix code**
  None

- **Initial setting**
  OFF; off

- **Example**
  ALARM ON;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

**LMTUP**

LMTUP

- **Function**
  Sets the upper limit line area on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMTUP</td>
<td>LMTUP(\Delta\ sw)</td>
<td>LMTUP?</td>
<td>LMTUP(\Delta\ sw) (sw = 0, 1)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 1, ON ; On
  - 0, OFF ; Off

- **Suffix code**
  None

- **Initial setting**
  OFF

- **Example**
  LMTUP ON;

---

**LMTLW**

LMTLW

- **Function**
  Sets the lower limit line area on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMTLW</td>
<td>LMTLW(\Delta\ sw)</td>
<td>LMTLW?</td>
<td>LMTLW(\Delta\ sw) (sw = 0, 1)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 1, ON ; On
  - 0, OFF ; Off

- **Suffix code**
  None

- **Initial setting**
  OFF

- **Example**
  LMTLW ON;
LMTUPD / LMTLWD
LMTUPD/LMTLWD  Upper / Lower Limit Line Data Input / Output

**Function**  Provide a method for transferring limit line data to or from a computer. Data format is decimal number(d) format.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMTUPD</td>
<td>LMTUPD#1#500,#d0,...,#dn</td>
<td>LMTUPD#1#500,?</td>
<td>d = Decimal data (ASCII code)</td>
</tr>
<tr>
<td>LMTLWD</td>
<td>LMLTLD#1#500,#d0,...,#dn</td>
<td>LMLTLD#1#500,?</td>
<td>[d0,d1,...,#dn]</td>
</tr>
</tbody>
</table>

**Value of #**  1 to 500 (Point number); Fixed.

**Value of d**  
- LOG : 0 to 3600
- Linear : 0 to 1800
- AM, FM : 0 to 3600
- QP MODE: 0 to 4000

**Example**  
- LMTUPD #1 \#500,\<2048,1248,200, \..........2430,\>
- LMTUPD #1 \#500,;  
- LMTLWD #1 \#500,\<2048,1248,200, \..........2430,\>
- LMTLWD #1 \#500,;
SECTION 4 DETAILED DESCRIPTION OF COMMANDS

WINDOW

WIN

WIN Window Function On / Off

Function Sets the Window function on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIN</td>
<td>WIN\sw</td>
<td>WIN?</td>
<td>WIN\sw sw = 0,1</td>
</tr>
</tbody>
</table>

Value of sw

1, ON : On
0, OFF : Off

Suffix code None
Initial setting OFF
Example WIN ON;

WINUP

WINUP Upper Window

Function Selects the upper window area.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINUP</td>
<td>WINUP</td>
<td>~ ~ ~</td>
<td>~ ~ ~ ~ ~</td>
</tr>
</tbody>
</table>

Example WINUP;
**WINLW**

WINLW **Lower Window**

- **Function**: Selects the lower window area.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINLW</td>
<td>WINLW</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Example**: WINLW;

---

**WINUZ**

WINUZ **Window Upper Zoom-in**

- **Function**: Changes the upper window area to the entire display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINUZ</td>
<td>WINUZ</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Example**: WINUZ;
**WINLZ**

**WINLZ** Window lower Zoom-in

*Function* Changes the lower window area to the entire display.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINLZ</td>
<td>WINLZ</td>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

*Example* WINLZ;

---

**WINT**

**WINT** Window Toggle

*Function* Toggles between the upper window area and lower window area.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINT</td>
<td>WINT(\Delta) sw</td>
<td>WINT?</td>
<td>WINT(\Delta) sw (sw = 0, 1)</td>
</tr>
</tbody>
</table>

*Value of sw* 1, ON : On 0, OFF : Off

*Suffix code* None

*Initial setting* OFF

*Example* WINT ON;
MEASUREMENT

dB Down

XDBDW

XDBDW X dB Down

Function Places left and right marker at X dB down from the reference maker.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBDW</td>
<td>XDBDWΔ l</td>
<td>XDBDW?</td>
<td>XDBDWΔ l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>l = 3 to 80 (0.1dB step)</td>
</tr>
</tbody>
</table>

- Value of l 3 to 80 (0.1dB step)
- Suffix code DB
- Initial setting OFF
- Example XDBDW 6DB;

XDBLW

XDBLW X dB Left Down

Function Places a marker at the point to the left of the reference maker that is X dB down.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBLW</td>
<td>XDBLWΔ l</td>
<td>XDBLW?</td>
<td>XDBLWΔ l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>l = 3 to 80 (0.1dB step)</td>
</tr>
</tbody>
</table>

- Value of l 3 to 80 (0.1dB step)
- Suffix code DB
- Initial setting OFF
- Example XDBLW 6DB;
**XDBRW**

**Function**
Places a marker at the point to the right of the reference marker that is X dB down.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBRW</td>
<td>XDBRW(\uparrow) l</td>
<td>XDBRW?</td>
<td>XDBRW(\uparrow) l (l = 3) to (80) (0.1dB step)</td>
</tr>
</tbody>
</table>

- **Value of l**: 3 to 80 (0.1dB step)
- **Suffix code**: DB
- **Initial setting**: OFF
- **Example**: XDBRW 6DB;

---

**XDBSEL**

**Function**
Selects the X dB marker display format.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBSEL</td>
<td>XDBSEL(\uparrow) sw</td>
<td>XDBSEL?</td>
<td>XDBSEL(\uparrow) sw (sw = REL, ABSR, ABSL)</td>
</tr>
</tbody>
</table>

- **Value of sw**: REL : Relative
  ABSR : Right Absolute
  ABSL : Left Absolute
- **Suffix code**: None
- **Initial setting**: REL
- **Example**: XDBSEL REL;
### XDBSGL

**XDBSGL**  
**Function** Single Sweep and X dB Measurement  
**Description** X dB down measurement is executed only once with sweep is finished.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBSGL</td>
<td>XDBSGL</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**  
XDBSGL;

### XDBCTN

**XDBCTN**  
**Function** Continuous Sweep and X dB Measurement.  
**Description** X dB down measurement is executed continuously, after each sweep.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBCTN</td>
<td>XDBCTN</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**  
XDBCTN;

### XDBEND

**XDBEND**  
**Function** Stop the X dB Measurement function  
**Description** Stops the X dB Measurements function.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBEND</td>
<td>XDBEND</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**  
XDBEND;
### XDBRF

**XDBRF**

**Function**

- **Return Frequency result of the X dB Measurement**
- *Returns the frequency result of the X dB Measurements.*

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBRF</td>
<td>- - - -</td>
<td>XDBRF?</td>
<td>XDBRF $\Delta f \ f = 0 \text{ to } 2.7\text{GHz}$</td>
</tr>
</tbody>
</table>

- **Value of $f$**: 0Hz to 2.7GHz
  - If XDBSEL; REL : Relative Value
  - ABSR : Right Absolute Value
  - ABSL : Left Absolute Value

- **Example**
  ```
  XDBRF?;
  ```

### XDBRA

**XDBRA**

**Function**

- **Return Amplitude result of the X dB Measurement**
- *Returns the amplitude result of the X dB Measurements.*

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDBRA</td>
<td>- - - -</td>
<td>XDBRA?</td>
<td>XDBRA $\Delta I$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I = -100\text{dBm to } +30\text{dBm(or dB)}$</td>
</tr>
</tbody>
</table>

- **Value of $I$**
  - -100dBm ~ +30dBm(or dB), 0.1 step
  - If XDBSEL; REL : Relative Value
  - ABSR : Right Absolute Value
  - ABSL : Left Absolute Value

- **Example**
  ```
  XDBRA?;
  ```
**Occupied Power Bandwidth Measurement**

**OBW**

*OBW*

**Occupied power BandWidth Measurement**

- **Function**: Calculates and displays occupied power bandwidth.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBW</td>
<td>OBW(\Delta) n</td>
<td>OBW?</td>
<td>OBW(\Delta) n (n = 10.0) to 99.8</td>
</tr>
</tbody>
</table>

- **Value of n**: 10.0 to 99.8 (0.1% step)
- **Suffix code**: None (%)
- **Initial setting**: 10
- **Example**: OBW 30;

---

**OBWR**

*OBWR*

**Occupied power BandWidth Measurement Value**

- **Function**: Returns occupied power bandwidth measurements value.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBWR</td>
<td>- - - - - - - -</td>
<td>OBWR?</td>
<td>OBWR(\Delta) f (f = 0) to 2.7GHz</td>
</tr>
</tbody>
</table>

- **Value of f**: 0Hz to 2.7GHz
- **Suffix code**: None : Hz(10^-0)  
  HZ : Hz(10^-0)  
  KHZ : kHz(10^-3)  
  MHZ : MHz(10^-6)  
  GHZ : GHz(10^-9)
- **Example**: OBWR?;
### Channel Power

**CHP**

**Channel Power**

- **Function**: Measuring channel power.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHP</td>
<td>CHP</td>
<td>CHP?</td>
<td>CHP(\triangle res)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\text{res} = xx.xx \text{dBm})</td>
</tr>
</tbody>
</table>

- **Suffix code**: None

- **Example**: CHP?;

- **Example**: CHP;

### CHPHZ

**Channel Power Spectral Density**

- **Function**: Read channel power spectral density

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHPHZ</td>
<td>- - - - -</td>
<td>CHPHZ?</td>
<td>CHPHZ(\triangle res)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\text{res} = xx.xx \text{dBm/Hz})</td>
</tr>
</tbody>
</table>

- **Suffix code**: None

- **Example**: CHPHZ?;
**CHPCB**

**CHPCB** Channel Bandwidth in Channel Power

- **Function**: Set channel bandwidth in channel power measurement.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHPCB</td>
<td>CHPCB(\Delta f)</td>
<td>CHPCB?</td>
<td>CHPCB(\Delta res)</td>
</tr>
<tr>
<td></td>
<td>CHPCB(\Delta a)</td>
<td></td>
<td>(res = 20\text{Hz} \sim 1\text{GHz})</td>
</tr>
</tbody>
</table>

- **Value of f**: 20 Hz \sim 1 GHz
- **Value of a**:
  - **OA**: Function Query (same as ?)
  - **UP**: Increment size is 1/10 of current span.
  - **DN**: Decrement size is 1/10 of current span.
- **Suffix code**:
  - **None**: Hz(10^0)
  - **HZ**: Hz(10^0)
  - **KHZ**: kHz(10^3)
  - **MHZ**: MHz(10^6)
  - **GHZ**: GHz(10^9)

- **Initial setting**: Value of f = 15 kHz
- **Example**:
  - CHPCB 2000;
  - CHPCB 1.23MHZ;
  - CHPCB?;
## Adjacent Channel Power

**ACP**

ACP

Adjacent Channel Power Ratio

- **Function**: Measuring adjacent channel power ratio.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACP</td>
<td>ACP</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: ACP;

## ACPC

ACPC

Carrier Channel Power

- **Function**: Read carrier channel power in ACP

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPC</td>
<td>- - - -</td>
<td>ACPC?</td>
<td>ACPC(\triangle ) res</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>res = xx.xx dBm</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: ACPC?;
### ACPL

**ACPL**
Lower Channel Power Ratio in ACP

- **Function**: Read lower channel power ratio in ACP measurement.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPL</td>
<td>- - - -</td>
<td>ACPL</td>
<td>ACPL△ res res = xx.xx dB</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: ACPL?;

### ACPU

**ACPU**
Upper Channel Power Ratio in ACP

- **Function**: Read upper channel power ratio in ACP measurement.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPU</td>
<td>- - - -</td>
<td>ACPU?</td>
<td>ACPU△ res res = xx.xx dB</td>
</tr>
</tbody>
</table>

- **Suffix code**: None
- **Example**: ACPU?;
ACPCB
ACPCB Channel Bandwidth in ACP
■ Function Set channel bandwidth in ACP measurement.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPCB</td>
<td>ACPCBΔ f</td>
<td>ACPCB?</td>
<td>ACPCBΔ res</td>
</tr>
<tr>
<td></td>
<td>ACPCBΔ a</td>
<td></td>
<td>res = 20Hz~1GHz</td>
</tr>
</tbody>
</table>

■ Value of f 20 Hz ~ 1 GHz
■ Value of a
  OA : Function Query (same as ?)
  UP : Increment size is 1/10 of current span.
  DN : Decrement size is 1/10 of current span.
■ Suffix code
  None : Hz(10^0)
  HZ : Hz(10^0)
  KHZ : kHz(10^3)
  MHZ : MHz(10^6)
  GHZ : GHz(10^9)
■ Initial setting Value of f = 15 kHz
■ Example ACPCB 2000;
             ACPCB 1.23MHZ;
             ACPCB?;
ACPSP

ACPSP Channel Space in ACP

- Function
  Set channel space in ACP measurement.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPSP</td>
<td>ACPSP Δ f</td>
<td>ACPSP?</td>
<td>ACPSP Δ res</td>
</tr>
<tr>
<td></td>
<td>ACPSP Δ a</td>
<td></td>
<td>res = 20Hz~1GHz</td>
</tr>
</tbody>
</table>

- Value of f: 20 Hz ~ 1 GHz
- Value of a:
  - OA: Function Query (same as ?)
  - UP: Increment size is 1/10 of current span.
  - DN: Decrement size is 1/10 of current span.

- Suffix code:
  - None: Hz(10^0)
  - HZ: Hz(10^0)
  - KHZ: kHz(10^3)
  - MHZ: MHz(10^6)
  - GHZ: GHz(10^9)

- Initial setting: Value of f = 20 kHz

- Example:
  - ACPSP 2000;
  - ACPSP 1.23MHZ;
  - ACPSP?;
Quasi Peak Mode (Option)

**QPM**
- **QPM**
  - **Function**: Quasi Peak Mode
  - **Selects quasi peak mode bandwidth.**

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPM</td>
<td>QPMΔ sw</td>
<td>QPM?</td>
<td>QPMΔ sw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sw = BNDB, BNDC, OFF</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - BNDB : RBW 9kHz
  - BNDC : RBW 120kHz
  - OFF : off

- **Suffix code** None
- **Initial setting** OFF
- **Example** QPM BNDB;
## Frequency Counter

**MKFC**

**MKFC**
- **Function**: Activates a frequency counter that counts the frequency of the normal marker.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKFC</td>
<td>MKFC(\Delta) sw</td>
<td>MKFC?</td>
<td>MKFC(\Delta) sw (\text{sw} = \text{OFF}, \text{freq.} = 0 \text{ to } 2.7GHz)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - ON : On
  - OFF : Off
  - 0 to 2.7GHz

- **Suffix code**: None
- **Initial setting**: OFF
- **Example**: MKFC ON;

## MKFCR

**MKFCR**

**MKFCR**
- **Function**: Specifies the resolution of the frequency counter.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKFCR</td>
<td>MKFCR(\Delta) f</td>
<td>MKFCR?</td>
<td>MKFCR(\Delta) f (f = 1, 10, 100, 1000) Transfer data with no unit of Hz</td>
</tr>
</tbody>
</table>

- **Value of f** 1, 10, 100, 1000
- **Suffix code**: None
- **Initial setting**: 1000
- **Example**: MKFCR 1000;
AUXILIARY

**DEMOD**

**Function**
Activates either AM or FM demodulation or turns the demodulation off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMOD</td>
<td>DEMOD(\Delta) sw</td>
<td>DEMOD?</td>
<td>DEMOD(\Delta) sw (sw = AM, FM, OFF)</td>
</tr>
</tbody>
</table>

- **Value of\(sw\):**
  - AM: AM demodulation
  - FM: FM demodulation
  - OFF: Off

- **Suffix code:**
  - NONE

- **Initial setting:**
  - OFF

- **Example:**
  - DEMOD AM;

**AUDIO**

**Function**
Selects Speaker on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIO</td>
<td>AUDIO(\Delta) sw</td>
<td>AUDIO?</td>
<td>AUDIO(\Delta) sw (sw = 0, 1)</td>
</tr>
</tbody>
</table>

- **Value of\(sw\):**
  - 1, ON: On
  - 0, OFF: Off

- **Suffix code:**
  - None

- **Initial setting:**
  - 0, OFF

- **Example:**
  - AUDIO ON;

**SQL**

**Function**
Adjusts squelch level.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>SQL(\Delta) n</td>
<td>SQL?</td>
<td>SQL(\Delta) n (n = 0) to 255</td>
</tr>
</tbody>
</table>

- **Value of\(n\):** 0 to 255 (1 step)

- **Suffix code:**
  - None

- **Initial setting:**
  - 115

- **Example:**
  - SQL 115;
### AUDIOVR

**AUDIOVR**

**Function**

Speaker Volume

Adjusts the volume of the speaker.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIOVR</td>
<td>AUDIOVR(\Delta n)</td>
<td>AUDIOVR?</td>
<td>AUDIOVR(\Delta n) (n = 0) to (7)</td>
</tr>
</tbody>
</table>

- **Value of** \(n\) \(0\) to \(7\) (1 step)
- **Suffix code** None
- **Example** AUDIOVR 5;

### AMS

**AMS**

**AM Scale**

**Function**

Set AM scale

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>AMS(\Delta sw)</td>
<td>AMS?</td>
<td>AMS(\Delta rs) (rs = 25.0, 20.0, 10.0, 5.0, 2.5, 0.0 = \text{not FM mode})</td>
</tr>
</tbody>
</table>

- **Value of** \(sw\) \(25, 20, 10, 5, 2.5;\)
- **Suffix code** None[%]
- **Initial setting** 25
- **Example** AMS 20;

### FMS

**FMS**

**FM Scale**

**Function**

Set FM scale

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMS</td>
<td>FMS(\Delta sw)</td>
<td>FMS?</td>
<td>FMS(\Delta rs) (rs = 25.0, 20.0, 10.0, 5.0, 2.5, 1.0, 0.0 = \text{not FM mode})</td>
</tr>
</tbody>
</table>

- **Value of** \(sw\) \(25, 20, 10, 5, 2.5, 1;\)
- **Suffix code** None[kHz]
- **Initial setting** 25
- **Example** FMS 20;
## PRESET

**PRST**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Executes preset. All instrument parameters are set to default values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRST</td>
<td>PRST</td>
<td>------</td>
<td>----------</td>
</tr>
</tbody>
</table>

**Example**

PRST;

## CALALL

**CALALL**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Executes all calibrations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALALL</td>
<td>CALALL</td>
<td>------</td>
<td>----------</td>
</tr>
</tbody>
</table>

**Example**

CALALL;
### PCAL

**Function**

Temperature Calibrations Executions On or Off

Initiates periodic temperature calibration execution.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCAL</td>
<td>PCAL(\triangle ) sw</td>
<td>PCAL?</td>
<td>PCAL(\triangle ) sw (sw = 0, 1)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - 1, ON : On
  - 0, OFF : Off

- **Suffix code**
  - None

- **Initial setting**
  - ON

- **Example**
  - PCAL ON;

### YIGCAL

**Function**

First Local Oscillator Calibration

Executes First local oscillator Calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>YIGCAL</td>
<td>YIGCAL</td>
<td>------</td>
<td>---------</td>
</tr>
</tbody>
</table>

- **Example**
  - YIGCAL;
### RCAL

**RCAL**

RBW Calibration

- **Function**: Executes RBW Calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCAL</td>
<td>RCAL</td>
<td>------</td>
<td>-----------</td>
</tr>
</tbody>
</table>

**Example**: `RCAL;`

### TMPCAL

**TMPCAL**

Temperature Calibration

- **Function**: Executes temperature Calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMPCAL</td>
<td>TMPCAL</td>
<td>------</td>
<td>-----------</td>
</tr>
</tbody>
</table>

**Example**: `TMPCAL;`
**LVLC**

LVLC Level Calibration

**Function**
Executes level calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVLC</td>
<td>LVLC</td>
<td>-------</td>
<td>---------</td>
</tr>
</tbody>
</table>

**Example**
LVLC;

---

**SPCAL**

SPCAL Span Calibration

**Function**
Executes span calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPCAL</td>
<td>SPCAL</td>
<td>-------</td>
<td>---------</td>
</tr>
</tbody>
</table>

**Example**
SPCAL;

---

**LAC**

LAC LOG Calibration

**Function**
Executes LOG amplifier calibration.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAC</td>
<td>LAC</td>
<td>-------</td>
<td>---------</td>
</tr>
</tbody>
</table>

**Example**
LAC;
### CALSIG

**CALSIG**

- **Function**: Calibration Signal On or Off
- **Description**: Turns the calibration signal on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALSIG</td>
<td>CALSIG(\Delta) sw</td>
<td>CALSIG?</td>
<td>CALSIG(\Delta) sw ( sw = 0, 1 )</td>
</tr>
</tbody>
</table>

- **Value of sw**: 1, ON : On  
  0, OFF : Off
- **Suffix code**: None
- **Initial setting**: ON
- **Example**: CALSIG ON;

### ACAL

**ACAL**

- **Function**: Amplitude Calibration
- **Description**: Turns amplitude calibration tables on or off.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACAL</td>
<td>ACAL(\Delta) sw</td>
<td>ACAL?</td>
<td>ACAL(\Delta) sw ( sw = ON, OFF )</td>
</tr>
</tbody>
</table>

- **Value of sw**: 1, ON : Amplitude Calibration Tables ON  
  0, OFF: Amplitude Calibration Tables OFF
- **Suffix code**: None
- **Initial setting**: ON
- **Example**: ACAL ON;
CONFIGURATION

PRINTER

HCOPY

HCOPY Hard Copy

* Function Prints hard copy.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCOPY</td>
<td>HCOPY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Example HCOPY;

HCPTYP

HCPTYP Hard Copy Type

* Function Sets hard copy type.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCPTYP</td>
<td>HCPTYP\sw</td>
<td>HCPTYP?</td>
<td>HCPTYP\sw \sw = PRT, CRD</td>
</tr>
</tbody>
</table>

* Value of sw PRT : Printer
CRD : PCMCIA Memory Card (Data format : BMP)

* Example HCPTYP PRT;
## CLOCK SET

### DATE

**Function**
Sets the built-in clock to the specified date.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>DATE() yy.mm.dd</td>
<td>DATE?</td>
<td>DATE() yy.mm.dd yy.mm.dd</td>
</tr>
</tbody>
</table>

- Value of mm: 01 to 12 (month)
- Value of dd: 01 to 31 (day)
- Suffix code: None
- Example: DATE 98.08.21;

### TIME

**Function**
Sets the built-in clock to the specified time.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>TIME() hh.mm.ss</td>
<td>TIME?</td>
<td>TIME() hh:mm:ss hh:mm:ss</td>
</tr>
</tbody>
</table>

- Value of hh: 00 to 23 (hour)
- Value of mm: 00 to 59 (minute)
- Value of ss: 00 to 59 (sec)
- Suffix code: None
- Example: TIME 09.11.22;
PCMCIA(option)

PCMCHK

PCMCHK PCMCIA Check

- Function Returns the PCMCIA memory Card status.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMCHK</td>
<td>- - - -</td>
<td>PCMCHK?</td>
<td>PCMCHK△sw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sw = - 2, - 1, 0, 1, 2</td>
</tr>
</tbody>
</table>

- Value of sw
  - 2 : No card
  - 1 : Not format
  0  : Status OK
  1  : Protected
  2  : Low or bad battery

- Example
  PCMCHK?;
REFERENCE CLOCK

REFLO

REFLO Reference Clock
■ Function Selects the reference clock.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFLO</td>
<td>REFLO(\Delta) sw</td>
<td>REFLO?</td>
<td>REFLO(\Delta) sw sw = INT, EXT</td>
</tr>
</tbody>
</table>

■ Value of sw
  INT : Internal clock
  EXT : External clock

■ Suffix code
  None

■ Initial setting
  INT

■ Example
  REFLO INT;
## GPIB COMMON COMMAND

### *CLS*

- **CLS**
- **Function**: Clear Status Command
- **Description**: Clears the status byte register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* CLS</td>
<td>* CLS</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example**: *CLS;

### *ESE*

- **ESE**
- **Function**: Standard Event Status Enable
- **Description**: Sets or clears the standard status enable register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ESE</td>
<td>* ESE(n)</td>
<td>-</td>
<td><strong>ESE(n)</strong> (n = 0) to 255</td>
</tr>
</tbody>
</table>

**Value of n**: 0 to 255
**Suffix code**: None
**Example**: *ESE 20;
*ESE?;
### * ESR?  
* ESR? Standard Event Status Register Query  

**Function**  
Returns the current value in the standard event status register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ESR</td>
<td>- - - - - -</td>
<td>* ESR?</td>
<td>* ESRₙ n = 0 to 255</td>
</tr>
</tbody>
</table>

- **Value of n**  
0 to 255  
- **Suffix code**  
None  
- **Example**  
* ESR?;

### * IDN?  
* IDN? Identification Query  

**Function**  
Return the model name, etc of the equipment.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* IDN</td>
<td>- - - - - -</td>
<td>* IDN?</td>
<td>IFR, 2398, serial, version</td>
</tr>
</tbody>
</table>

- **Suffix code**  
None  
- **Example**  
* IDN?;
**OPC**

- **OPC**
  - **Function**: Sets bit 0 in the standard event status register when all pending select device operations have been completed.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* OPC</td>
<td>* OPC</td>
<td>- - - - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- - - - -</td>
<td></td>
</tr>
</tbody>
</table>

- **Example** * OPC;

---

**OPC?**

- **OPC?**
  - **Function**: Sets the output queue to 1 to generate a MAV summary message when all pending select device operations have been completed.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* OPC?</td>
<td>- - - - -</td>
<td>* OPC?</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Example** * OPC?;
### * RST

**Reset Command**

- **Function**: Resets the device.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* RST</td>
<td>* RST</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

**Example**

* RST;

### * SRE

**Service Request Enable Command**

- **Function**: Sets the bits in the service request enable register.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* SRE</td>
<td>* SREΔ n</td>
<td>* SRE?</td>
<td>* SREΔ n n = 0 to 255</td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 255
- **Suffix code**: None
- **Example**
  
  * SRE 1; 
  * SRE?;
**STB?**

Returns Status Byte Command

**Function**

Returns the current values of the status bytes including the MSS bit.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>* STB</td>
<td>- - - - - - - -</td>
<td>* STB?</td>
<td>* STBₜ ₙ ₙ = 0 to 255</td>
</tr>
</tbody>
</table>

**Value of n**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit weight</th>
<th>Bit name</th>
<th>Condition of status byte register</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>128</td>
<td>- - - - - -</td>
<td>0 = Not used</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>MSS</td>
<td>0 = Service not requested</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Service requested</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>ESB</td>
<td>0 = Event status not generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Event status generated</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>MAV</td>
<td>0 = No data in output queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Data in output queue</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>ESB2</td>
<td>0 = Event status not generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Event status generated</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>- - - - - -</td>
<td>0 = Not used</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>- - - - - -</td>
<td>0 = Not used</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>- - - - - -</td>
<td>0 = Not used</td>
</tr>
</tbody>
</table>

**Example**

* STB?;
OTHERS

ESE2

Event Status Enable (End)

- **Function**: Allows the End Event Status Enable Register to select which bit in the corresponding Event Register causes a TRUE ESB summary message bit 3 when set.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESE2</td>
<td>ESE2Δ n</td>
<td>ESE2?</td>
<td>ESE2Δ n \ n = 0 to 255</td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 255: Represents the sum of the bit-weighted values.
- **Suffix code**: None
- **Example**: ESE2 1; ESE2?;

ESR2?

Event Status Register (End) Query

- **Function**: Allows the sum of binary-weighted event bit values of the End Event Status Register to be read out by converting them to decimal. After readout, the End Event status Register is reset to 0.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESR2?</td>
<td>- - - - - - - -</td>
<td>ESR2?</td>
<td>ESR2Δ n \ n = 0 to 255</td>
</tr>
</tbody>
</table>

- **Value of n**: 0 to 255
- **Suffix code**: None
- **Example**: ESR2?;
**QRYTYP**

QRYTYP

Query Response Type

- **Function**
  - Sets query response type.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>QRYTYP</td>
<td>QRYTYP(\Delta) sw</td>
<td>QRYTYP?</td>
<td>QRYTYP(\Delta) sw (sw = ECO, NEC)</td>
</tr>
</tbody>
</table>

- Value of \(sw\) 
  - **ECO**: Query response type is echo.
  - **NEC**: Query response type is no echo.
- Initial setting: **NEC**
- Example: QRYTYP NEC;

---

**ERR**

ERR

Error Number

- **Function**
  - Returns the error number of the current function.
  - Error buffer size: 256

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR</td>
<td>- - - - - - - -</td>
<td>ERR?</td>
<td>ERR(\Delta) (n = \times 0(x: 0 \text{ to } 9))</td>
</tr>
</tbody>
</table>

- Value of \(n\) 
  - \(n\) : 000 to 999 (Refer to the Appendix - Error Code)
- Example: ERR?;

---

**ECHO**

ECHO

RS-232C echo

- **Function**
  - Echo received characters to the host on RS-232C

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECHO</td>
<td>ECHO(\Delta) sw</td>
<td>ECHO?</td>
<td>ECHO(\Delta) (res = 1, 0)</td>
</tr>
</tbody>
</table>

- Value of \(sw\) 
  - 1, ON : echo on
  - 0, OFF : echo off
- Initial setting: **ON**
- Example: ECHO ON;
TRACKING GENERATOR (option)

TGEN
TGEN Tracking Generator Power
- Function Sets the output power of tracking generator

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
</table>
| TGEN   | TGEN\(\Delta\) sw | TGEN? | TGEN\(\Delta\) rsw  
\[ rsw = \text{ON, OFF} \] |

- Value of sw  
  ON : On  
  OFF : Off  
  OA : Function Query (same as ?)
- Suffix code None
- Initial setting OFF : Off
- Example TGEN ON;

TGLEV
TGLEV Tracking Generator Output Level
- Function Sets the output level of tracking generator

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
</table>
| TGLEV  | TGLEV\(\Delta\) l  
TGLEV\(\Delta\) a | TGLEV? | TGLEV\(\Delta\) l  
\[ l = -70.0 \sim 0.0\text{dBm} \]  
current output level |

- Value of l Value from -70.0dBm to 0.0dBm (step 0.1dBm)
- Value of a  
  UP : Increment level (step 1dBm)  
  DN : Decrement level (step 1dBm)  
  OA : Function Query (same as ?)
- Suffix code None : dBm  
DBM : dBm
- Initial setting \( l = 0.0\text{dBm} \)
- Example TGLEV - 12.5DBM;  
TGLEV 0DBM;
**TGNORM**

TGNORM Tracking Generator Normalize

**Function**
Sets the normalize function of tracking generator

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGNORM</td>
<td>TGNORM(\triangle sw)</td>
<td>TGNORM?</td>
<td>TGNORM(\triangle rs)</td>
</tr>
</tbody>
</table>

- **Value of sw**
  - ON : On
  - OFF : Off
  - OA : Function Query (same as ?)

- **Suffix code** None
- **Initial setting** OFF : Off
- **Example** TGNORM ON;

---

**TGAFC**

TGAFC Tracking Generator Auto Frequency Calibration

**Function**
Executes frequency calibration for tracking generator automatically

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGAFC</td>
<td>TGAFC</td>
<td>- - - -</td>
<td>- - - - - -</td>
</tr>
</tbody>
</table>

**Example** TGAFC;
TGMFC

TGMFC

Function

Tracks frequency calibration for the tracking generator manually.

<table>
<thead>
<tr>
<th>Header</th>
<th>Program command</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGMFC</td>
<td>TGMFC Δf</td>
<td>TGMFC?</td>
<td>TGMFC Δf</td>
</tr>
<tr>
<td></td>
<td>TGMFC Δa</td>
<td></td>
<td>f = offset frequency</td>
</tr>
</tbody>
</table>

- **Value of f**: Offset frequency, -3MHz to 3MHz
- **Value of a**: UP: Increment offset frequency f (step 20 Hz), DN: Decrement offset frequency f (step 20 Hz), OA: Function Query (same as ?)
- **Suffix code**: None: Hz, HZ: Hz
- **Initial setting**: f = 0Hz
- **Example**: TGMFC - 12HZ;
  TGMFC UP;
SECTION 5
STATUS STRUCTURE

This section describes the device-status reporting and its data structure defined by
the IEEE488.2 when GPIB interface bus is used. This section also describes the
synchronization techniques between a controller and device.
These functions are used to control a device from an external controller using the
GPIB interface bus. Most of these functions can also be used to control a device
from an external controller using the RS-232C interface.

<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE488.2 standard status model ......................................................... 5-3</td>
</tr>
<tr>
<td>Status byte(STB) register ................................................................. 5-5</td>
</tr>
<tr>
<td>ESB and MAV summary messages ............................................................. 5-5</td>
</tr>
<tr>
<td>Device-dependent summary messages ...................................................... 5-6</td>
</tr>
<tr>
<td>Reading and clearing the STB register .................................................... 5-7</td>
</tr>
<tr>
<td>Service request(SRQ) enabling operation ................................................ 5-8</td>
</tr>
<tr>
<td>Standard event status register .............................................................. 5-9</td>
</tr>
<tr>
<td>Bit definition of standard event status register ...................................... 5-9</td>
</tr>
<tr>
<td>Reading, writing, and clearing the standard event status register .......... 5-10</td>
</tr>
<tr>
<td>Reading, writing, and clearing the standard event status enable register .... 5-10</td>
</tr>
<tr>
<td>Extended event status register ............................................................. 5-11</td>
</tr>
<tr>
<td>Bit definition of END event status register .......................................... 5-12</td>
</tr>
<tr>
<td>Reading, writing, and clearing the extended event status register .......... 5-13</td>
</tr>
<tr>
<td>Reading, writing, and clearing the extended event status enable register .... 5-13</td>
</tr>
</tbody>
</table>
SECTION 5 STATUS STRUCTURE

<BLANK>
SECTION 5
STATUS STRUCTURE

The Status Byte(STB) sent to the controller is based on the IEEE488.1 standard. The bits comprising the STB are called status summary messages because they represent a summary of the current data in registers and queues.

IEEE488.2 Standard Status Model

The diagram below shows the standard model for the status data structures stipulated in the IEEE488.2 standard.

Standard Status Model Diagram
In the status model, IEEE488.1 status bytes are used for the lowest grade status. This status byte is composed of seven summary message bits from the higher grade status structure. To create these summary message bits, the status data structure is composed of two types of register and models.

<table>
<thead>
<tr>
<th>Register model</th>
<th>Queue model</th>
</tr>
</thead>
<tbody>
<tr>
<td>The register model consists of two registers used for recording events and conditions encountered by a device. These two registers are the Event Status Register and Event Status Enable Register. When the results of the AND operation of both register contents are other than 0, the corresponding bit of the status bit becomes 1. In other cases, the corresponding bit becomes 0. When the result of their Logical OR is 1, the summary message bit also becomes 1. If the Logical OR result is 0, the summary message bit also becomes 0.</td>
<td>The queue in the queue model is used to sequentially record the waiting status values or information. If the queue is not empty, the queue structure summary message becomes 1. If the queue is empty, the message becomes 0.</td>
</tr>
</tbody>
</table>

In IEEE488.2, there are three standard models for the status data structure. Two are register models and one is a queue model based on the register model and queue model described above. The three standard models are:
1. Standard Event Status Register and Standard Event Status Enable Register
2. Status Byte Register and Service Request Enable Register
3. Output Queue

<table>
<thead>
<tr>
<th>Standard Event Status Register</th>
<th>Status Byte Register</th>
<th>Output Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Standard Event Status Register has the same structure as the previously described register model. In this register, the bits for eight types of standard events encountered by a device are set as follows: ① Power on ② User request ③ Command error ④ Execution error ⑤ Device- dependent error ⑥ Query error ⑦ Request for bus control right ⑧ Operation complete The Logical OR output bit is represented by Status Byte Register bit 5 (DIO6) as a summary message for the Event Status Bit (ESB).</td>
<td>The Status Byte Register is a register in which the RQS bit and the seven summary message bits from the status data structure can be set. This register is used together with the Service Request Enable Register. When the results of the OR operation of both register contents are other than 0, SRQ becomes ON. To indicate this, bit 6 of the Status Byte Register (DIO7) is reserved by the system as the RQS bit. The RQS bit is used to indicate that there is a service request for the external controller. The mechanism of SRQ conforms to the IEEE488.1 standard.</td>
<td>The Output Queue has the structure of the queue model described above. Status Byte Register bit 4 (DIO5) is set as a summary message for Message Available (MAV) to indicate that there is data in the output buffer.</td>
</tr>
</tbody>
</table>
Status Byte (STB) Register

The STB register consists of the STB and RQS (or MSS) messages of the device.

ESB and MAV summary messages

This paragraph describes the ESB and MAV summary messages.

(1) ESB summary message
The ESB (Event Summary Bit) is a message defined by IEEE488.2 which uses bit 5 of the STB register. When the setting permits events to occur, the ESB summary message bit becomes 1 if any one of the events recorded in the Standard Status Register becomes 1. Conversely, the ESB summary message bit becomes 0 if one of the recorded events occurs, even if events are set to occur.

This bit becomes 0 when the ESR register is read by the *ESR? query or when it is cleared by the *CLS command.

(2) MAV summary message
The MAV (Message Available) summary bit is a message defined by IEEE488.2 which uses bit 4 of the STB register. This bit indicates whether the output queue is empty. The MAV summary message bit is set to 1 when a device is ready to receive a request for a response message from the controller. When the output queue is empty, this bit is set to 0. This message is used to synchronize the information exchange with the controller. For example, this message is available when, after the controller sends a query command to a device, the controller waits until MAV becomes 1. While the controller is waiting for a response from the device, other jobs can be processed. Reading the Output Queue without first checking MAV will cause all system bus operations to be delayed until the device responds.
Device-dependent summary messages

As shown in the diagram below, the 2398 does not use bits 0, 1, 2 and 7, and it uses bit 3 as the summary bit of the Event Status Register.

Service Request generated
(only when the GPB interface bus is used)
Reading and clearing the STB register.

The STB register can be read using serial polling or the *STB? common query. The IEEE488.1 STB message can be read by either method, but the value sent to bit 6(position) is different for each method. The STB register contents can be cleared using the *CLS command.

(1) Reading by serial polling (only when the GPIB interface bus is used)
   The IEEE488.1 serial polling allows the device to return a 7-bit status byte and an RQS message bit which conforms to IEEE488.1. The value of the status byte is not changed by serial polling. The device sets the RQS message to 0 immediately after being polled.

(2) Reading by the *STB? common query
   The *STB common query requires the devices to send the contents of the STB register and the integer format response messages, including the MSS (Master Summary Status) summary message. Therefore, except for bit 6, which represents the MSS summary message, the response to *STB? is identical to that of serial polling.

(3) Definition of MSS (Master Summary Message)
   MSS indicates that there is at least one cause for a service request. The MSS message is represented at bit 6 response to an *STB? Query, but it is not produced as a response to serial polling. It should not be taken as part of the status byte specified by IEEE488.1.
   MSS is configured by the over all logical OR in which the STB register and SRQ enable(SRE) register are combined.

(4) Clearing the STB register using the *CLS common command
   The *CLS common command clears all status data structures as well as the summary messages corresponding to them.
   The *CLS command does not affect the setting in the Enable Register.
**Service Request (SRQ) Enabling Operation**

Bits 0 to 7 of the Service Request Enable Register (SRE) determine which bit of the corresponding STB register can generate SRQ.

The bits in the Service Request Enable Register correspond to the bits in the Status Byte Register. If a bit in the Status Byte Register corresponding to an enabled bit in the Service Request Enable Register is set to 1, the device makes a service request to the controller with the RQS bit set to 1.

![Diagram of Service Request (SRQ) Enabling Operation](image)

(1) **Reading the SRE register**

The contents of the SRE register are read using the *SRE?* Common query. The response message to this query is an integer from 0 to 255 which is the sum of the bit digit weighted values in the SRE register.

(2) **Updating the SRE register**

The SRE register is written using the *SRE* common command. An integer from 0 to 255 is assigned as a parameter to set the SRE register bit to 0 or 1. The value of bit 6 is ignored.
Standard Event Status Register

Bit definition of Standard Event Status Register

The diagram below shows the operation of the Standard Event Status Register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Power on (PON- Power on)</td>
<td>A transition from power- off to power- on occurred during the power- up procedure.</td>
</tr>
<tr>
<td>6</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Command error (CME- Command Error)</td>
<td>An illegal program message or a misspelled command was received.</td>
</tr>
<tr>
<td>4</td>
<td>Execution error (EXE- Execution Error)</td>
<td>A legal but unexecutable program message was received</td>
</tr>
<tr>
<td>3</td>
<td>Device- dependent error (DDE- Device- dependent Error)</td>
<td>An error not caused by CME, EXE, or QYE occurred (parameter error, etc.).</td>
</tr>
<tr>
<td>2</td>
<td>Query error (QYE- Query Error)</td>
<td>An attempt was made to read data in the output queue when it was empty. Or, the data in the output queue was lost before it was read</td>
</tr>
<tr>
<td>1</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Operation complete (OPC- Operation Complete)</td>
<td>This bit becomes 1 when this instrument has Processed the *OPC command.</td>
</tr>
</tbody>
</table>
### Reading, writing, and clearing the Standard Event Status Register

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading</strong></td>
<td>The register is read using the <em>ESR?</em> command query. The register is cleared after being read. The response message is integer format data with the binary weight added to the event bit and the sum converted to decimal.</td>
</tr>
<tr>
<td><strong>Writing</strong></td>
<td>With the exception of clearing, data cannot be written to the register from outside.</td>
</tr>
</tbody>
</table>
| **Clearing** | The register is cleared when:  

  1. A *CLS command is received.  

  2. The power is turned on Bit 7 is set to ON, and the other bits are cleared to 0.  

  3. An event is read for the *ESR? query command. |

### Reading, writing, and clearing the Standard Event Status Enable Register

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading</strong></td>
<td>The register is read using the *ESE? command. The response message is integer format data with the binary weight added to the event bit and the sum converted to decimal.</td>
</tr>
<tr>
<td><strong>Writing</strong></td>
<td>The register is written using the *ESE common command.</td>
</tr>
</tbody>
</table>
| **Clearing** | The register is cleared when:  

  1. An *ESE command with a data value of 0 is received.  

  2. The power is turned on.  

The Standard Event Enable Register is not affected when:  

  1. The device clear function status of IEEE488.1 is changed.  

  2. A *RST common command is received.  

  3. A *CLS common command is received. |
Extended Event Status Register

For the 2398 bits 7, 2, 1, and 0 are unused. Bit 3 is assigned to the END summary bit as the status-summary bit supplied by the extended register model as shown below.
Bit definition of END Event Status Register

The diagram below shows the operation and event- bit names of the END Event Status Register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Not used</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>Not used</td>
<td>Not used</td>
</tr>
<tr>
<td>5</td>
<td>Not used</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>Measurement completed</td>
<td>Calculation processing for measurements (Peak search, OBW, X dB down, Marker noise, Frequency counter, Limit pass/fail) has been completed.</td>
</tr>
<tr>
<td>3</td>
<td>AUTO SET completed</td>
<td>AUTO SET has been completed.</td>
</tr>
<tr>
<td>2</td>
<td>Averaging completed</td>
<td>Sweeping according to the specified AVERAGE number has been completed.</td>
</tr>
<tr>
<td>1</td>
<td>Calibration completed</td>
<td>RBW CAL, Power on CAL, All CAL, Temp CAL, Span CAL, Level CAL, or LOG CAL has been completed.</td>
</tr>
<tr>
<td>0</td>
<td>Sweep completed</td>
<td>A single sweep has been completed or is standby</td>
</tr>
</tbody>
</table>
### Reading, writing, and clearing the Extended Event Status Register

<table>
<thead>
<tr>
<th>Reading</th>
<th>The ESR? query is used to read the register. The register is cleared after being read. The response message is integer-format data with the binary weight added to the event bit and the sum converted to decimal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing</td>
<td>With the exception of clearing, data cannot be written to the register from outside.</td>
</tr>
</tbody>
</table>
| Clearing  | The register is cleared when:  
1. A *CLS command is received.  
2. The power is turned on.  
3. An event is read for the ESR? query command.                                                                                           |

### Reading, writing, and clearing the Extended Status Enable Register

<table>
<thead>
<tr>
<th>Reading</th>
<th>The ESE? query is used to read the register. The response message is integer-format data with the binary weight added to the event bit and sum converted to decimals.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing</td>
<td>The ESE2 program command is used to write the register. Because bits 0 to 7 of the registers are weighted with values 1, 2, 4, 8, 16, 32, 64, and 128, respectively, the write data is transmitted as integer-format data that is the sum of the required bit digits selected from the weighted value.</td>
</tr>
</tbody>
</table>
| Clearing  | The register is cleared when:  
1. An ESE2 program command with a data value of 0 is received.  
2. The power is turned on.  
3. The Extended Event Status Enable register is not affected when:  
4. The device clear function status of IEEE488.1 is changed.  
5. A *RST common command is received.  
6. A *CLS common command is received.                                                                                                           |
SECTION 5 STATUS STRUCTURE

<BLANK>
SECTION 6
EXAMPLE CODES

This section shows some example codes to transmit the message on the bus between a personal computer and 2398 via GPIB

TABLE OF CONTENTS

Frequency and Level at the marker point .................................................. 6- 3
Delta Marker Measurement ................................................................. 6- 5
Frequency Bandwidth ........................................................................... 6- 7
Occupied Bandwidth Measurement ....................................................... 6- 9
Marker Noise Measurement ................................................................... 6-11
Saving Data ............................................................................................ 6-13
Recalling Data ......................................................................................... 6-14
Get Trace Data ....................................................................................... 6-15
Pass / Fail Check .................................................................................... 6-17
Frequency and Level Measurement

Sets the normal marker on the peak point at the current waveform and measures the frequency and level on the normal marker.

1. Set
   a. Span: 1MHz
   b. Center Frequency: 1300 MHz
   c. Reference Level: -10dBm
   d. VBW, RBW, Input Attenuator: Auto
   e. Log 10 dB scale, Unit: dBm
   f. Sweep Time: 50ms

2. Measuring
   a. Peak Search
   b. Read the frequency and the amplitude at the peak point

// Frequency and Level measurement at the marker point

#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>

int ud;
char DataBuf[80]; // Set Input message buffer size
char SpollByte;

void Send(char *buf)
{
    ibwt(ud, buf, (long)strlen(buf));
    if (ibsta & ERR) printf("ibwt Error\n");
}

void Receive(void)
{
    // Is data on 2398 buffer to read ?
    ibwait(ud, (TIMO | RQS));
    if (ibsta & (ERR | TIMO)) printf("ibwait Error\n");

    ibsp(ud, &SpollByte);
    if (ibsta & ERR) printf("ibsp Error\n");
    if (SpollByte != 0x50) printf("2398 Polling Error\n");

    // read data.
    ibrd(ud, DataBuf, 80L);
    DataBuf[lbcnd-1] = \0;
    if (ibsta & ERR) printf("ibrd Error\n");
}

void InitGPIOB(void)
{
    // Initialize GPIB bus and 2398
    ud = ibdev(0, 7, 0, T10s, 1, 0); // GPIB initialization, set 2398 to address 7
    if (ud < 0) printf("2398 device open error\n");
}
ibclr(ud);
if(ibsta & ERR) { printf("ibclr error\n"); exit(1); }
else printf("Init Ok\n");
}

void main(void)
{
  printf("<<<(Frequency and Level measurement)>>\n");
  InitGPIB();
  Send("*CLS;*SRE 16; "); // Set 2398 to its initial state for programming
  // span: 1MHz, Center Frequency: 1300 MHz, Reference Level: - 10dBm
  Send("*SP 1MHZ;CF 1300MHZ;RL - 10 DBM; ");
  Send("AUTOCL; "); // VBW, RBW, Sweep Time, Input Attenuator: Auto
  Send("LG 10 DB; "); // Log 10 dB scale
  Send("AUNITS DBM; "); // dBm unit
  Send("ST MAN;ST 50MS;*OPC?; ");
  Receive(); // Waiting the commands completed
  Send("MKPK; "); // Peak Search
  Send("MKA?; "); // Marker Amplitude Query 
  Receive(); // Read the Amplitude value
  printf("Amplitude = %s\n",DataBuf);
  Send("MKN7; "); // Marker Frequency Query 
  Receive(); // Read the Frequency value
  printf("Frequency = %s\n",DataBuf);
} // the end of main.
Delta Marker Measurement

Measuring the difference value of the frequency, amplitude between the normal and the delta marker, and 1/delta.

1. Set  a. Center Frequency: 500 MHz  
   b. Span: 500 kHz  
   c. Reference Level: -20 dBm  
   d. VBW: 30 kHz  
   e. RBW: 10 kHz  
   f. Sweep Time: 100 ms  
   g. Input Attenuator: Auto  
   h. Log 10 dB scale, Unit: dBm  

2. Measuring  a. Peak Search  
   b. Marker Frequency to Center Frequency  
   c. Marker Level to Reference Level  
   d. Delta Marker : Peak Point. Normal Marker: 500.050MHz  
   e. Read the Normal Marker frequency.  
   f. Read the difference between the Normal and the Delta Marker  
   g. Read 1/Delta

// -----------------------------
// Delta Marker measurement
// -----------------------------

#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>

int ud;
char DataBuf[80]; // Set Input message buffer size
char SpoilByte;

void Send(char *buf)
{
    ibwt(ud, buf, (long)strlen(buf));
    if (ibsta & ERR) printf("ibwt Error\n");
}

void Receive(void)
{
    // Is data on 2398 buffer to read ?
    ibwait(ud, (TIMO | RQS));
    if (ibsta & (ERR|TIMO)) printf("ibwait Error\n");

    ibsp(ud, &SpoilByte);
    if (ibsta & ERR) printf("ibsp Error\n");
    if (SpoilByte != 0x50) printf("2398 Polling Error\n");

    // read data.
    ibrd(ud, DataBuf, 80L);
    DataBuf[ibcnt- 1] = \0;
    if (ibsta & ERR) printf("ibrd Error\n");
}
void InitGPIB(void)
{
    // Initialize GPIB bus and 2398
    ud = ibdev(0,7,0,TI0s,1,0); // GPIB initialization, set 2398 to address 7
    if(ud<0) printf("2398 device open error\n");

    ibclr(ud);
    if(ibsta & ERR) { printf("ibc1r error\n"); exit(1); } else printf("Init Ok\n");
}

void main(void)
{
    printf("<<<Delta Marker measurement>>\n");

    InitGPIB();

    Send("*CLS;*SRE 16;*"); // Set 2398 to its initial state for programming

    Send("CF 500 MHz;SP 500 KHZ;*"); // Center Frequency 500 MHz Span 500 kHz

    Send("VB MAN;VB 30 KHZ;*"); // VBW Manual, VBW 30 kHz

    Send("RB MAN;RB 10 KHZ;*"); // RBW Manual RBW 10 kHz

    Send("ST MAN;ST 100 MS;*"); // Sweep Time Manual, Sweep Time 100ms

    Send("AT AUTO;*"); // Input Attenuator Auto

    Send("RL - 20 DBM;*"); // Reference Level - 20 dBm

    Send("LG 10 DB;*"); // Log 10 dB scale

    Send("AUNITS DBM;*OPC?;*"); // dBm unit

    Receive(); // Waiting the commands completed

    Send("MKPK;*"); // Peak Search

    Send("MKCF;*OPC?;*"); // Marker Frequency - Center Frequency

    Receive(); // Waiting the commands completed

    Send("MKRL;*OPC?;*"); // Marker Level - Reference Level

    Receive(); // Waiting the commands completed

    Send("MKD;*"); // Delta Mark : Peak point, Normal Marker : Peak point

    Send("MKN 500.050 MHz;*"); // Delta Mark : Peak point, Normal Marker : 500.050 MHz

    Send("MKN?;*"); // Read the normal marker frequency

    Receive();

    printf("Normal Marker Frequency = %s\n",DataBuf);

    Send("MKF?;*"); // Read the amplitude difference between the Delta and the Normal marker

    Receive();

    printf("Delta Amplitude = %s\n",DataBuf);

    Send("MKTDF?;*"); // Read the difference frequency

    Receive();

    printf("Delta Frequency = %s\n",DataBuf);

    Send("MKDTF;*"); // Set 1/Delta

    Send("MKTF?;*"); // Read 1/Delta

    Receive();

    printf("1/Delta = %s\n",DataBuf);

    // the end of main
Frequency Bandwidth

Searches the X dB point from the normal marker and measures X dB frequency bandwidth. (X is 6 dB on this example code.)

1. Set
   a. Center frequency: 100 MHz
   b. Span: 500 kHz
   c. Reference Level: -10 dBm
   d. VBW: 10 kHz
   e. RBW: 30 kHz,
   f. Sweep Time: Auto
   g. Input Attenuator: Auto
   h. Log 10 dB scale, Unit: dBm

2. Measuring
   a. Peak Search
   b. Marker Frequency to Center Frequency
   c. Marker Level to Reference Level
   d. Single Sweep
   e. X dB down format: Relative
   f. Set 6 dB down point from the normal marker
   g. Read 6 dB frequency bandwidth
   h. Stop X dB down
   i. Continuous Sweep

//-----------------------------------------------
// Frequency Bandwidth measurement
//-----------------------------------------------

#include <windows.h>
#include "Decl-32.h"
#include <stdio.h>

int ud;
char DataBuf[80]; // Set Input message buffer size

void Send(char *buf)
{
    ibwt(ud, buf, (long)strlen(buf));
    if (lsta & ERR) printf("ibwt Error\n");
}

void Receive(void)
{
    // read data.
    ibrd(ud, DataBuf, 80L);
    DataBuf[lbcnt-1] = '\0';
    if (lsta & ERR) printf("ibrd Error\n");
}

void InitGPIB(void)
\{  
  // Initialize GPIB bus and 2398
  ud = libdev(0,7,0,TI0s,1,0); // GPIB initialization, set 2398 to address 7
  if(ud<0) printf("2398 device open error\n");

  ibclr(ud);
  if(ibsta & ERR) { printf("ibclr error\n"); exit(1); }
  else printf("Init Ok\n");
\}

void main(void)
{
  printf("<<<<Frequency Bandwidth measurement>>>>\n");

  InitGPIB();

  Send("*CLS;"); // Set 2398 to its initial state for programming

  Send("*CF 100 MHZ;"); // Center Frequency 100 MHz
  Send("*SP 500 KHZ;"); // Span 500 kHz

  Send("*RL - 10 DBM;"); // Reference Level - 10dBm
  Send("*AT AUTO;ST AUTO;"); // Input Attenuator Auto, Sweep Time auto
  Send("*LG 10 DB; AUNITS DBM;"); // Log 10 db scale, dBm unit

  Send("*VB MAN;VB 10 KHZ;"); // VBW Manual, VBW 10 kHz
  Send("*RB MAN;RB 30 KHZ;*OPC?;"); // RBW Manual, RBW 30 kHz
  Receive(); // Waiting the commands completed

  Send("*MKPK;"); // Peak Search

  Send("*MKCF:*OPC?;"); // Marker Frequency - Center Frequency
  Receive(); // Waiting the commands completed

  Send("*MKRL:*OPC?;"); // Marker Amplitude - Reference Level
  Receive(); // Waiting the commands completed

  Send("*TRGSWP SNG;"); // Single Sweep for measuring

  Send("*XDBSEL REL;"); // X dB data display format : Relative
  Send("*XDBW 6DB:*OPC?;"); // 6dB down, Left and Right down from the Normal Marker.
  Receive(); // Waiting the commands completed

  Send("*XDBRF?;"); // Read X dB down Frequency Bandwidth
  Receive();
  printf("Frequency Bandwidth = %s\n",DataBuf);

  Send("*XDBEND;"); // Stop X dB down measurement.

  Send("*TRGSWP CNT;"); // Continuous Sweep

\} // the end of main
Occupied Power Bandwidth Measurement

Sets the normal marker on the carrier- frequency of the occupied band center frequency, and calculates OBW (Occupied Power Bandwidth).

1. Set
   a. Detection mode: SAMPLE
   b. Center frequency: 100 MHz
   c. Span: 2 MHz
   d. Input Attenuator: Auto
   e. Unit: dBm, Log 10 dB scale,
   f. Reference Level: -25 dBm
   g. RBW: 10 kHz
   h. VBW: 1 kHz
   i. Sweep Time: Auto

2. Measuring
   a. Peak Search
   b. Marker Frequency to Center Frequency
   c. Set OBW 50%
   d. Waiting for OBW calculation completed
   e. Read OBW

// ---------------------------------------------------------
// OBW measurement
// ---------------------------------------------------------

#include <windows.h>
#include "Dec- 32.h"
#include <stdio.h>
int ud;
char DataBuf[80]; // Set Input message buffer size
char SpollByte;

void Send(char *buf)
{
  ibwrt(ud, buf, (long)strlen(buf));
  if (ibsta & ERR) printf("ibwrt Error\n");
}

void Receive(void)
{
  ibrd(ud, DataBuf, 80L);
  DataBuf[ibcntl- 1] = \0';
  if (ibsta & ERR) printf("ibrd Error\n");
}

void InitGPIB(void)
{
  // Initialize GPIB bus and Z398
  ud = libdev(0,7,0,T10s,1,0); // GPIB initialization, set 2398 to address 7
  if(ud<0) printf("2398 device open error\n");

  ibclr(ud);
  if(ibsta & ERR) { printf("ibclr error\n"); exit(1); }
  else printf("Init OK\n");
}
void main(void)
{
    printf("<<<OBW measurement>>\n");

    InitGPIB();

    Send("*CLS;*"); // Set 2398 to its initial state for programming
    Send("DET SAM;*"); // Sets the detection mode to SAMPLE
    Send("CF 100 MHZ;*"); // Center Frequency 100 MHz
    Send("SP 2 MHZ;*"); // Span 2 MHz
    Send("AT AUTO;ST AUTO;*"); // Input Attenuator Auto, Sweep time auto

    Send("AUNITS DBM;*"); // dBm unit
    Send("LG 10 DB;*"); // Log 10 dB scale
    Send("RL - 25 DBM;*"); // Reference Level - 25 dBm
    Send("RB MAN;RB 10 KHZ;*"); // RBW Manual, RBW 10 kHz

    Send("VB MAN;VB 1 KHZ;OPC?;*"); // VBW Manual, VBW 1 kHz
    Receive(); // Waiting the commands completed

    Send("MKPK;*"); // Peak Search
    Send("MKCF;OPC?;*"); // Marker Frequency - Center Frequency
                  Receive(); // Waiting the commands completed

    Send("OBW 50;OPC?;*"); // set OBW 50%
    Receive(); // Waiting the commands completed

    Send("OBWR?;*"); // Query OBW?
    Receive(); // Read OBW
    printf("OBW = %s\n",DataBuf); // the end of main
Marker Noise Measurement

Sets the reference marker on the signal, and the normal marker on the noise, then measures Marker Noise.

1. Set
   a. Center frequency: 300 MHz
   b. Span: 1 MHz
   c. Reference Level: -20dBm
   d. Input Attenuator: Auto
   e. Log 10 dB scale, Unit: dBm
   f. RBW: 10 kHz
   g. VBW: Auto
   h. Sweep Time: 50ms

2. Measuring
   a. Delta Marker: Peak point, Normal Marker: 300.100 MHz
   b. Read Marker Noise

    /***************************************************************************/
    // Marker Noise measurement
    /***************************************************************************/
    #include <windows.h>
    #include "Decl- 32.h"
    #include <stdio.h>

    int ud;
    char DataBuf[80]; // Set Input message buffer size
    char SpollByte;

    void Send(char *buf)
    {
      lbwt(ud, buf, (long)strlen(buf));
      if (lsta & ERR) printf("lbwt Error\n");
    }

    void Receive(void)
    {
      lbrd(ud, DataBuf, 80L);
      DataBuf[lbcnt- 1] = '\0';
      if (lsta & ERR) printf("lbrd Error\n");
    }

    void InitGPIB(void)
    {
      /***************************************************************************/
      // Initialize GPIB bus and 2398
      /***************************************************************************/
      ud = lbdev(0,7,0,T10s,1,0); // GPIB initialization, set 2398 to address 7
      if(ud<0) printf("2398 device open error\n");

      lbcrl(ud);
      if(lsta & ERR) { printf("lbcrl error\n"); exit(1); }
      else printf("Init OK\n");
    }
void main(void)
{
    printf("<<<Marker Noise measurement>>>\n");
    InitGPIB();
    Send("*CLS;"); // Set 2398 to its initial state for programming
    Send("CF 300 MHZ;"); // Center Frequency 300 MHz
    Send("SP 1 MHZ;"); // Span 1 MHz
    Send("RL - 20 DBM;*"); // Reference Level - 20dBm
    Send("AT AUTO;*"); // Input attenuator Auto
    Send("LG 10 DB;*"); // Log 10 dB scale
    Send("AUNITS DB;*"); // dBm Scale
    Send("RB MAN;RB 10 KHZ;*"); // RBW Manual, RBW 10 kHz
    Send("VB AUTO;"); // VBW Auto
    Send("ST MAN;ST 50 MS;OPC?;*"); // Sweep Time Manual, Sweep Time 50ms
    Receive(); // Waiting the commands completedRec
    Send("MKPK;*"); // Peak Search
    Send("MKRL;*OPC?;*"); // Marker Level - Reference Level
    Receive(); // Waiting the commands completed
    Send("MKD;*"); // Delta Mark : Peak point, Normal Marker : Peak point
    Send("MKN 300.100MHZ;*"); // Delta Mark : Peak point, Normal Marker : 300.100MHZ
    Send("MKNOISE ON;*OPC?;*");
    Receive(); // Waiting the commands completed
    Send("MKNOISE ?;*");
    Receive(); // Read Marker Noise
    printf("Marker Noise = %s\n",DataBuf);
} // the end of main
Saving Data(option)

Saves the current system status to PCMCIA card.

```c
#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>
#include <stdlib.h>

int ud;
char DataBuf[80]; // Set Input message buffer size
char SpollByte;

void Send(char *buf)
{
    lbwt(ud, buf, (long)strlen(buf));
    if (lbsta & ERR) printf("lbwt Error\n");
}

void Receive(void)
{
    lbrd(ud, DataBuf, 80L);
    DataBuf[lbcntl- 1] = \0;
    if (lbcsta & ERR) printf("lbrd Error\n");
}

void InitGPIB(void)
{
    // Initialize GPIB bus and 2398
    ud = lbdv(0,7,0,T10s,1,0); // GPIB initialization, set 2398 to address 7
    if(ud<0) printf("2398 device open error\n");
    lbcrl(ud);
    if(lbsta & ERR) { printf("lbcrl error\n"); exit(1); } else printf("Init OK\n");
}

void main(void)
{
    printf("<<<Save the current status to PCMCIA>>>\n");
    InitGPIB();
    Send("*CLS;*"); // Set 2398 to its initial state for programming
    Send("TITLE MEAS1");
    Send("PCMCHK;*");
    Receive();
    if(atol(DataBuf) == 0) {
        Send("SVMS:*OPC?;*");
        Receive(); // Waiting the commands completed
    } else
        printf("PCMCIA ERR/\n");
}

// the end of main
```
Recalling Data (option)

Recalls the system status from PCMCIA card.

```c
#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>
#include <stdlib.h>

int ud;
char DataBuf[80]; // Set input message buffer size
char SpoilByte;

void Send(char *buf)
{
    ibwt(ud, buf, (long)strlen(buf));
    if (ibsta & ERR) printf("ibwt Error\n");
}

void Receive(void)
{
    ibrd(ud, DataBuf, 80L);
    DataBuf[ibcntl - 1] = '\0';
    if (ibsta & ERR) printf("ibrd Error\n");
}

void InitGPIB(void)
{
    // Initialize GPIB bus and 2398
    ud = ibdev(0,7,0,T10s,1,0); // GPIB initialization, set 2398 to address 7
    if(ud<0) printf("2398 device open error\n");
    ibclr(ud);
    if(ibsta & ERR) { printf("ibclr error\n"); exit(1); }
    else printf("Init OK\n");
}

void main(void)
{
    printf("<<<Recall the current status from PCMCIA>>>\n");
    InitGPIB();
    Send("*CLS;*"); // Set 2398 to its initial state for programming
    Send("PCMCHK?;*");
    Receive();
    if(atoi(DataBuf)== 0) {
        Send("RCM MEAS1.STS:*OPC?*"); //Recall "MEAS1.STS" from PCMCIA
        Receive(); //Waiting the commands completed
    } else {
        printf("PCMCIA ERR\n");
    }
}
```

// the end of main
Get Trace Data

Get all trace data from 2398.

```c
#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>
#include <stdlib.h>

int ud;
unsigned char DataBuf[4096]; // Set input message buffer size
       // [3600,2555,...]⇒ “3600,” : 5byte 5byte×500 + ... ⇒ about 4096
char SpollByte;

void Send(char *buf)
{
  ibwt(ud, buf, (long)strlen(buf));
  if (ibsta & ERR) printf("ibwt Erron\n");
}

void Receive(void)
{
  // Is data on 2398 buffer to read?
  ibwait(ud, (TIMO | ROS));
  if (ibsta & (ERR[TIMO])) printf("ibwait Erron\n");

  lbrsp(ud, &SpollByte);
  if (ibsta & ERR) printf("lbrsp Erron\n");
  if (SpollByte != 0x50) printf("2398 Polling Erron\n");

  // read data.
  lbrd(ud, DataBuf, 4096L);
  DataBuf[lbcnt- 1] = \0;
  if (ibsta & ERR) printf("lbrd Erron\n");
}

void InitGPIB(void)
{
  // Initialize GPIB bus and 2398
  ud = libdev(0,7,0,TIOS,1,0); // GPIB initialization, set 2398 to address 7
  if(ud<0) printf("2398 device open error\n");

  lbcrl(ud);
  if(ibsta & ERR) { printf("lbcrl error\n"); exit(1); } else printf("Init OK\n");
}

void main(void)
{
  int i,j,n;
  unsigned char c;
  char TempBuf[10];
  short int TraceData[510];
  unsigned char Title[30];
  short int Temp16BitInt;

  printf("<<<Get all the Trace Data>>>\n");
  InitGPIB();
}
Send("*CLS;*SRE 16;"); // Set 2398 to its initial state for programming

// For the binary type
Send("TDF BIN; ");

// For the decimal type
Send("TDF DEC ;");

Send("TRAALL? ;");
Receive();

for(i=0; DataBuf[i] != ' '; i++)
{
    Title[i] = DataBuf[i];
}

Title[i] = '0';
printf("- - %s- - \n", Title); // display Title
n = i + 1;

// For the binary type
for(i=0; i <= 499; i++)
{
    Temp16BitInt = DataBuf[n++]; // upper byte
    Temp16BitInt <<= 8;
    Temp16BitInt += DataBuf[n++]; // lower byte

    TraceData[i] = Temp16BitInt;
    printf("TRACE[%d] = %d\n", i, TraceData[i]);
}

// For the decimal type
for(i=0; i <= 499; i++)
{
    j=0;

do{
        c = DataBuf[j++];
        TempBuff[++] = c;
    }while( isdigit(c) );

    TempBuff[j] = '0';
    TraceData[i] = atoi(TempBuff);
    printf("TRACE[%d] = %d\n", i, TraceData[i]);
}

}// the end of main
Pass / Fail Check

Check PASS or FAIL by comparing the current waveform with the upper limit line or the lower limit line.

```c
// Pass / Fail Check
#include <windows.h>
#include "Decl- 32.h"
#include <stdio.h>
int ud;
char DataBuf[80]; // Set Input message buffer size

void SendBuf(char *buf)
{
    lbwrtd(buf, (long)strlen(buf));
    if (lsta & ERR) printf("lbwrtd Error\n");
}

void Receive(void)
{
    lbrd(ud, DataBuf, 80L);
    DataBuf[lbcrd- 1] = '\0';
    if (lsta & ERR) printf("lbrd Error\n");
}

void InitGPIB(void)
{
    // Initialize GPIB bus and 2398
    ud = ibdev(0,7,0,T10s,1,0); // GPIB initialization, set 2398 to address 7
    if (ud<0) printf("2398 device open error\n");

    lbcrl(ud);
    if (lsta & ERR) { printf("lbcrl error\n"); exit(1); } else printf("Init OK\n");
}

void main(void)
{
    printf("<<<Pass / Fail Check>>>\n");

    InitGPIB();

    Send("*CLS;"); // Set 2398 to its initial state for programming
    Send("RCLMT 3;"); // When RCLMT 3 is completed, The configuration is replaced by the data to have saved.
    Send("LMTUP ON;"); // upper limit on
    Send("LMTLW ON;"); // lower limit on

    Send("LMTPC ON:*OPC?;"); // pass/fail check RUN, check LMTPC completed.
    Receive(); // Waiting the commands completed

    Send("LMTPC?;"); // Query ? Pass/Fail
    Receive(); // Read the result pass/fail check
    printf("Pass/Fail Result = %s\n",DataBuf);
}
```

```c
  ```
## APPENDIX - PROGRAMMING COMMANDS

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACAL</td>
<td>Amplitude calibration tables on or off</td>
<td>4- 78</td>
</tr>
<tr>
<td>ACP</td>
<td>Adjacent Channel Power Ratio</td>
<td>4- 66</td>
</tr>
<tr>
<td>ACPC</td>
<td>Carrier Channel Power</td>
<td>4- 66</td>
</tr>
<tr>
<td>ACPCB</td>
<td>Channel Bandwidth in ACP</td>
<td>4- 68</td>
</tr>
<tr>
<td>ACPL</td>
<td>Lower Channel Power Ratio in ACP</td>
<td>4- 67</td>
</tr>
<tr>
<td>ACPSP</td>
<td>Channel Space in ACP</td>
<td>4- 69</td>
</tr>
<tr>
<td>ACPU</td>
<td>Upper Channel Power Ratio in ACP</td>
<td>4- 67</td>
</tr>
<tr>
<td>ALARM</td>
<td>Sets alarm when limit pass/fall check</td>
<td>4- 53</td>
</tr>
<tr>
<td>AMB</td>
<td>Trace A minus Trace B</td>
<td>4- 38</td>
</tr>
<tr>
<td>AMBPL</td>
<td>Trace A minus Trace B plus display line.</td>
<td>4- 39</td>
</tr>
<tr>
<td>AMS</td>
<td>Set AM scale</td>
<td>4- 73</td>
</tr>
<tr>
<td>ANNOT</td>
<td>Turns the display annotation on or off</td>
<td>4- 32</td>
</tr>
<tr>
<td>APB</td>
<td>Trace A plus trace B</td>
<td>4- 39</td>
</tr>
<tr>
<td>AT</td>
<td>Sets the amount of attenuation</td>
<td>4- 12</td>
</tr>
<tr>
<td>AUDIO</td>
<td>Selects Speaker on or off.</td>
<td>4- 72</td>
</tr>
<tr>
<td>AUDIOVR</td>
<td>Adjusts the volume of the speaker.</td>
<td>4- 73</td>
</tr>
<tr>
<td>AUNITS</td>
<td>Sets the absolute amplitude units.</td>
<td>4- 13</td>
</tr>
<tr>
<td>AUTOCPL</td>
<td>Sets the auto coupling mode.</td>
<td>4- 26</td>
</tr>
<tr>
<td>AUTOSET</td>
<td>Auto set</td>
<td>4- 44</td>
</tr>
<tr>
<td>AVG</td>
<td>Trace average on or off.</td>
<td>4- 42</td>
</tr>
<tr>
<td>AVGCR</td>
<td>Sets the averaging rate (number of sweep repetitions).</td>
<td>4- 42</td>
</tr>
<tr>
<td>AVGCRYL</td>
<td>Set averaging cycle on or off.</td>
<td>4- 43</td>
</tr>
<tr>
<td>AVGS</td>
<td>Set averaging cycle stop, continue or reset.</td>
<td>4- 43</td>
</tr>
<tr>
<td>AXB</td>
<td>Trace A exchange trace B</td>
<td>4- 40</td>
</tr>
<tr>
<td>BLANK</td>
<td>Blanks the trace from the display.</td>
<td>4- 34</td>
</tr>
<tr>
<td>BML</td>
<td>Trace B Minus display line.</td>
<td>4- 38</td>
</tr>
<tr>
<td>CALALL</td>
<td>Executes all calibrations.</td>
<td>4- 74</td>
</tr>
<tr>
<td>CALSIG</td>
<td>Sets the calibration signal on or off.</td>
<td>4- 78</td>
</tr>
<tr>
<td>CF</td>
<td>Sets the center frequency</td>
<td>4- 4</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>CHP</td>
<td>Channel Power</td>
<td>4- 64</td>
</tr>
<tr>
<td>CHPCB</td>
<td>Channel Bandwidth in Channel Power</td>
<td>4- 65</td>
</tr>
<tr>
<td>CHPHZ</td>
<td>Channel Power Spectral Density</td>
<td>4- 64</td>
</tr>
<tr>
<td>CLEW</td>
<td>Sets the chosen trace to clear- write mode.</td>
<td>4- 33</td>
</tr>
<tr>
<td>*CLS</td>
<td>Clears the status byte register.</td>
<td>4- 83</td>
</tr>
<tr>
<td>CNTRST</td>
<td>Controls the display contrast.</td>
<td>4- 31</td>
</tr>
<tr>
<td>DATE</td>
<td>Sets the built- in clock to the specified date.</td>
<td>4- 80</td>
</tr>
<tr>
<td>DEMOD</td>
<td>Activates either AM or FM demodulation.</td>
<td>4- 72</td>
</tr>
<tr>
<td>DET</td>
<td>Selects the detection mode.</td>
<td>4- 41</td>
</tr>
<tr>
<td>DL</td>
<td>Sets the display line.</td>
<td>4- 29</td>
</tr>
<tr>
<td>ECHO</td>
<td>RS- 232C echo</td>
<td>4- 89</td>
</tr>
<tr>
<td>ERR</td>
<td>Error number</td>
<td>4- 89</td>
</tr>
<tr>
<td>*ESE</td>
<td>Sets or clears the standard status enable register.</td>
<td>4- 83</td>
</tr>
<tr>
<td>ESE2</td>
<td>Event Status Enable (End)</td>
<td>4- 88</td>
</tr>
<tr>
<td>*ESR?</td>
<td>Return the value in the standard event status register.</td>
<td>4- 84</td>
</tr>
<tr>
<td>ESR2?</td>
<td>Event Status Register (End) Query</td>
<td>4- 88</td>
</tr>
<tr>
<td>FA</td>
<td>Sets the start frequency</td>
<td>4-  5</td>
</tr>
<tr>
<td>FB</td>
<td>Sets the stop frequency</td>
<td>4-  6</td>
</tr>
<tr>
<td>FMS</td>
<td>Set FM scale</td>
<td>4- 73</td>
</tr>
<tr>
<td>FS</td>
<td>Selects full frequency span</td>
<td>4-  9</td>
</tr>
<tr>
<td>GRAT</td>
<td>Turns the display graticule on or off.</td>
<td>4- 32</td>
</tr>
<tr>
<td>HCOPY</td>
<td>Prints hard copy.</td>
<td>4- 79</td>
</tr>
<tr>
<td>HCPTYP</td>
<td>Sets hard copy type.</td>
<td>4- 79</td>
</tr>
<tr>
<td>*IDN?</td>
<td>Return the model, software version NO. of the equipment.</td>
<td>4- 84</td>
</tr>
<tr>
<td>LAC</td>
<td>Executes LOG calibrations.</td>
<td>4- 77</td>
</tr>
<tr>
<td>LG</td>
<td>Selects logarithmic amplitude mode &amp; dB scale.</td>
<td>4- 12</td>
</tr>
<tr>
<td>LMTLW</td>
<td>Sets the lower limit line area on or off.</td>
<td>4- 54</td>
</tr>
<tr>
<td>LMTPC</td>
<td>Sets the limit line function on or off.</td>
<td>4- 53</td>
</tr>
<tr>
<td>LMTUP</td>
<td>Sets the upper limit line area on or off.</td>
<td>4- 54</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>LMTPD/LMTLWD</td>
<td>Upper / lower limit line data input / output.</td>
<td>4- 55</td>
</tr>
<tr>
<td>LN</td>
<td>Selects a linear amplitude mode.</td>
<td>4- 13</td>
</tr>
<tr>
<td>LVLC</td>
<td>Executes level calibration.</td>
<td>4- 77</td>
</tr>
<tr>
<td>MKA</td>
<td>Returns the amplitude of the active maker.</td>
<td>4- 16</td>
</tr>
<tr>
<td>MKCF</td>
<td>Marker to center frequency.</td>
<td>4- 18</td>
</tr>
<tr>
<td>MKD</td>
<td>Places an delta marker on the normal marker position.</td>
<td>4- 17</td>
</tr>
<tr>
<td>MKDTF</td>
<td>Marker 1/delta</td>
<td>4- 16</td>
</tr>
<tr>
<td>MKFC</td>
<td>Activates frequency counter for the normal marker.</td>
<td>4- 71</td>
</tr>
<tr>
<td>MKFCR</td>
<td>Specifies the resolution of the frequency counter.</td>
<td>4- 71</td>
</tr>
<tr>
<td>MKN</td>
<td>Places an active marker on the specified frequency.</td>
<td>4- 15</td>
</tr>
<tr>
<td>MKNOISE</td>
<td>Marker noise.</td>
<td>4- 22</td>
</tr>
<tr>
<td>MKOFF</td>
<td>Turns off the marker mode.</td>
<td>4- 17</td>
</tr>
<tr>
<td>MKPK</td>
<td>Peak Search.</td>
<td>4- 18</td>
</tr>
<tr>
<td>MKRL</td>
<td>Marker to Reference level.</td>
<td>4- 19</td>
</tr>
<tr>
<td>MKSP</td>
<td>Marker delta to span.</td>
<td>4- 19</td>
</tr>
<tr>
<td>MKSS</td>
<td>Marker to center frequency step size.</td>
<td>4- 20</td>
</tr>
<tr>
<td>MKTF</td>
<td>Read the marker frequency or time</td>
<td>4- 17</td>
</tr>
<tr>
<td>MKTRACK</td>
<td>Signal track.</td>
<td>4- 21</td>
</tr>
<tr>
<td>MKZI</td>
<td>Marker zoom- in.</td>
<td>4- 20</td>
</tr>
<tr>
<td>MKZO</td>
<td>Marker zoom- out.</td>
<td>4- 21</td>
</tr>
<tr>
<td>MXMH</td>
<td>Maximum Hold.</td>
<td>4- 34</td>
</tr>
<tr>
<td>OBW</td>
<td>Calculates and displays the occupied power bandwidth.</td>
<td>4- 63</td>
</tr>
<tr>
<td>OBWR</td>
<td>Returns occupied power bandwidth measurements value.</td>
<td>4- 63</td>
</tr>
<tr>
<td>*OPC</td>
<td>Operation complete command</td>
<td>4- 85</td>
</tr>
<tr>
<td>*OPC?</td>
<td>Operation complete query</td>
<td>4- 85</td>
</tr>
<tr>
<td>PCAL</td>
<td>Sets temperature calibration execution.</td>
<td>4- 75</td>
</tr>
<tr>
<td>PCMCHK</td>
<td>Returns the PCMCIA memory card status.</td>
<td>4- 81</td>
</tr>
<tr>
<td>PRST</td>
<td>Executes preset.</td>
<td>4- 74</td>
</tr>
<tr>
<td>QPM</td>
<td>Selects quasi peak mode bandwidth.</td>
<td>4- 70</td>
</tr>
<tr>
<td>QRYTYP</td>
<td>Sets query response type.</td>
<td>4- 89</td>
</tr>
<tr>
<td>RB</td>
<td>Sets the resolution bandwidth.</td>
<td>4- 26</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>RCAL</td>
<td>Executes RBW calibration.</td>
<td></td>
</tr>
<tr>
<td>RCLMT</td>
<td>Recalls previously saved mask data to the display.</td>
<td></td>
</tr>
<tr>
<td>RCM</td>
<td>Recalls the date from a memory card.</td>
<td></td>
</tr>
<tr>
<td>RCS</td>
<td>Recalls to the display a saved instrument state.</td>
<td></td>
</tr>
<tr>
<td>RCTRA</td>
<td>Recalls trace A from internal register.</td>
<td></td>
</tr>
<tr>
<td>RCTRIB</td>
<td>Recalls trace B from internal register.</td>
<td></td>
</tr>
<tr>
<td>REFLO</td>
<td>Selects the reference clock.</td>
<td></td>
</tr>
<tr>
<td>RL</td>
<td>Sets the reference level.</td>
<td></td>
</tr>
<tr>
<td>RLO</td>
<td>Reference level offset</td>
<td></td>
</tr>
<tr>
<td>*RST</td>
<td>Resets the device.</td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td>Sets the frequency span</td>
<td></td>
</tr>
<tr>
<td>SPCAL</td>
<td>Executes span calibration.</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>Adjusts squelch level.</td>
<td></td>
</tr>
<tr>
<td>*SRE</td>
<td>Sets the bits in the service request enable register.</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td>Sets the center frequency step size</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>Sets the sweep time.</td>
<td></td>
</tr>
<tr>
<td>*STB?</td>
<td>Returns the status bytes including the MSS bit.</td>
<td></td>
</tr>
<tr>
<td>SVLCK</td>
<td>Save lock on or off.</td>
<td></td>
</tr>
<tr>
<td>SVLMT</td>
<td>Saves the mask data into internal register.</td>
<td></td>
</tr>
<tr>
<td>SVMLMT</td>
<td>Saves the mask data into memory card.</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>Saves the current status data into memory card.</td>
<td></td>
</tr>
<tr>
<td>SVMTRA</td>
<td>Saves the Trace A into memory card.</td>
<td></td>
</tr>
<tr>
<td>SVMTRB</td>
<td>Saves the Trace B into memory card.</td>
<td></td>
</tr>
<tr>
<td>SVS</td>
<td>Saves the current status data into internal register.</td>
<td></td>
</tr>
<tr>
<td>SVTRA</td>
<td>Saves the Trace A in the specified trace register.</td>
<td></td>
</tr>
<tr>
<td>SVTRB</td>
<td>Saves the Trace B in the specified trace register.</td>
<td></td>
</tr>
<tr>
<td>TDLY</td>
<td>Sets the trigger delay time.</td>
<td></td>
</tr>
<tr>
<td>TDF</td>
<td>Trace data format</td>
<td></td>
</tr>
<tr>
<td>TGAFC</td>
<td>Tracking Generator Auto Frequency Calibration</td>
<td></td>
</tr>
<tr>
<td>TGEN</td>
<td>Tracking Generator Power</td>
<td></td>
</tr>
<tr>
<td>TGLEV</td>
<td>Tracking Generator Output Level</td>
<td></td>
</tr>
<tr>
<td>TGMCFC</td>
<td>Tracking Generator Manual Frequency Calibration</td>
<td></td>
</tr>
<tr>
<td>TGNORM</td>
<td>Tracking Generator Normalize</td>
<td></td>
</tr>
<tr>
<td>TF</td>
<td>Selects the trigger filter.</td>
<td></td>
</tr>
<tr>
<td>TH</td>
<td>Sets the threshold.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>TIME</td>
<td>Sets the built-in clock to the specified time.</td>
<td>4-80</td>
</tr>
<tr>
<td>TITLE</td>
<td>Screen title entry.</td>
<td>4-31</td>
</tr>
<tr>
<td>TLV</td>
<td>Sets the trigger level.</td>
<td>4-24</td>
</tr>
<tr>
<td>TM</td>
<td>Sets the trigger switch and trigger source.</td>
<td>4-23</td>
</tr>
<tr>
<td>TMPCAL</td>
<td>Executes temperature calibration.</td>
<td>4-76</td>
</tr>
<tr>
<td>TRA/TRB</td>
<td>Trace data input/output</td>
<td>4-35</td>
</tr>
<tr>
<td>TRAALL/TRBALL</td>
<td>Trace all data output</td>
<td>4-36</td>
</tr>
<tr>
<td>TRGSWP</td>
<td>Selects the continuous-sweep or the single-sweep mode.</td>
<td>4-23</td>
</tr>
<tr>
<td>TRSTS</td>
<td>Sets the trace status.</td>
<td>4-37</td>
</tr>
<tr>
<td>VB</td>
<td>Sets the video bandwidth.</td>
<td>4-27</td>
</tr>
<tr>
<td>VIEW</td>
<td>Displays the current contents of the selected trace.</td>
<td>4-33</td>
</tr>
<tr>
<td>WIN</td>
<td>Sets the Window function on or off.</td>
<td>4-56</td>
</tr>
<tr>
<td>WNLW</td>
<td>Selects the lower window area.</td>
<td>4-57</td>
</tr>
<tr>
<td>WNLZ</td>
<td>Changes the lower window area to the entire display.</td>
<td>4-58</td>
</tr>
<tr>
<td>WINT</td>
<td>Toggles the display between the upper window area and lower window area.</td>
<td>4-58</td>
</tr>
<tr>
<td>WINUP</td>
<td>Selects the upper window area.</td>
<td>4-56</td>
</tr>
<tr>
<td>WINUZ</td>
<td>Changes the upper window area to the entire display.</td>
<td>4-57</td>
</tr>
<tr>
<td>XDBCTN</td>
<td>Continuous sweep and X dB measurement.</td>
<td>4-61</td>
</tr>
<tr>
<td>XDBDW</td>
<td>Places left and right marker at X dB down.</td>
<td>4-59</td>
</tr>
<tr>
<td>XDBEND</td>
<td>Stop the X dB measurement function</td>
<td>4-61</td>
</tr>
<tr>
<td>XDBLW</td>
<td>Places the marker at X dB down to the left.</td>
<td>4-59</td>
</tr>
<tr>
<td>XDBRA</td>
<td>Returns the amplitude result of the X dB Measurements.</td>
<td>4-62</td>
</tr>
<tr>
<td>XDBRF</td>
<td>Returns the frequency result of the X dB Measurements.</td>
<td>4-62</td>
</tr>
<tr>
<td>XDBRW</td>
<td>Places the marker at X dB down to the right.</td>
<td>4-60</td>
</tr>
<tr>
<td>XDBSEL</td>
<td>Selects the X dB marker display format.</td>
<td>4-60</td>
</tr>
<tr>
<td>XDBSGL</td>
<td>Single sweep and X dB measurement.</td>
<td>4-61</td>
</tr>
<tr>
<td>YIGCAL</td>
<td>Executes First local oscillator Calibration.</td>
<td>4-75</td>
</tr>
<tr>
<td>ZI</td>
<td>Zoom-in (Changes to 1/2 the previous span)</td>
<td>4-10</td>
</tr>
<tr>
<td>ZO</td>
<td>Zoom-out (Changes to two times of the previous span)</td>
<td>4-10</td>
</tr>
<tr>
<td>ZS</td>
<td>Selects zero frequency span</td>
<td>4-9</td>
</tr>
</tbody>
</table>
APPENDIX-PROGRAMMING COMMANDS

〈BLANK〉
## APPENDIX - ERROR CODE

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>ERROR DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Data out of range.</td>
</tr>
<tr>
<td>101</td>
<td>Center frequency out of range.</td>
</tr>
<tr>
<td>102</td>
<td>Start frequency out of range.</td>
</tr>
<tr>
<td>103</td>
<td>Stop frequency out of range.</td>
</tr>
<tr>
<td>104</td>
<td>Center frequency step size out of range.</td>
</tr>
<tr>
<td>105</td>
<td>Span frequency out of range.</td>
</tr>
<tr>
<td>106</td>
<td>Reference level out of range</td>
</tr>
<tr>
<td>107</td>
<td>Attenuator level out of range.</td>
</tr>
<tr>
<td>108</td>
<td>Marker function out of range.</td>
</tr>
<tr>
<td>111</td>
<td>RBW out of range.</td>
</tr>
<tr>
<td>112</td>
<td>VBW out of range.</td>
</tr>
<tr>
<td>113</td>
<td>Sweep time out of range.</td>
</tr>
<tr>
<td>114</td>
<td>Contrast level out of range.</td>
</tr>
<tr>
<td>118</td>
<td>Threshold level error.</td>
</tr>
<tr>
<td>119</td>
<td>&quot;LG&quot; data out of range.</td>
</tr>
<tr>
<td>120</td>
<td>ESE data out of range.</td>
</tr>
<tr>
<td>121</td>
<td>ESE2 data out of range.</td>
</tr>
<tr>
<td>122</td>
<td>SRE data out of range.</td>
</tr>
<tr>
<td>123</td>
<td>Trigger Level data out of range.</td>
</tr>
<tr>
<td>124</td>
<td>Trigger Level, Trigger Source is not Video.</td>
</tr>
<tr>
<td>125</td>
<td>Trig Delay data out of range.</td>
</tr>
<tr>
<td>126</td>
<td>Trig Delay, Span is not Zero.</td>
</tr>
<tr>
<td>127</td>
<td>Average Count out of range.</td>
</tr>
<tr>
<td>128</td>
<td>Audio Level out of range.</td>
</tr>
<tr>
<td>129</td>
<td>Squelch Level out of range.</td>
</tr>
<tr>
<td>130</td>
<td>Date out of range.</td>
</tr>
<tr>
<td>131</td>
<td>Time out of range.</td>
</tr>
<tr>
<td>132</td>
<td>Save internal error.</td>
</tr>
<tr>
<td>133</td>
<td>Save external error.</td>
</tr>
<tr>
<td>134</td>
<td>Recall internal error.</td>
</tr>
<tr>
<td>135</td>
<td>Recall external error.</td>
</tr>
<tr>
<td>136</td>
<td>PCMCIA error.</td>
</tr>
<tr>
<td>137</td>
<td>Limit error.</td>
</tr>
<tr>
<td>138</td>
<td>dB down error.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>ERROR DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>139</td>
<td>OBW error.</td>
</tr>
<tr>
<td>140</td>
<td>TRA out of range.</td>
</tr>
<tr>
<td>141</td>
<td>TRB out of range.</td>
</tr>
<tr>
<td>142</td>
<td>Delta marker function error.</td>
</tr>
<tr>
<td>143</td>
<td>Normal marker function error.</td>
</tr>
<tr>
<td>144</td>
<td>Display Line error.</td>
</tr>
<tr>
<td>145</td>
<td>Marker Counter Resolution out of Range.</td>
</tr>
<tr>
<td>146</td>
<td>Noise Marker error.</td>
</tr>
<tr>
<td>147</td>
<td>Printer not connected or not responded.</td>
</tr>
<tr>
<td>148</td>
<td>Printer off-line or paper empty.</td>
</tr>
<tr>
<td>149</td>
<td>PCMCIA option not installed.</td>
</tr>
<tr>
<td>150</td>
<td>Auto SetUp, signal not found.</td>
</tr>
<tr>
<td>151</td>
<td>Peak search error.</td>
</tr>
<tr>
<td>152</td>
<td>RBW CAL Fail.</td>
</tr>
<tr>
<td>153</td>
<td>PWR On CAL Fail.</td>
</tr>
<tr>
<td>154</td>
<td>Temp. CAL Fail.</td>
</tr>
<tr>
<td>155</td>
<td>Tracking Generator is not ON</td>
</tr>
<tr>
<td>156</td>
<td>Tracking Generator output level error</td>
</tr>
<tr>
<td>157</td>
<td>Tracking Generator manual frequency CAL. offset frequency range error</td>
</tr>
<tr>
<td>158</td>
<td>Span Cal Fail</td>
</tr>
<tr>
<td>159</td>
<td>Level Cal Fail</td>
</tr>
<tr>
<td>160</td>
<td>Log Cal Fail</td>
</tr>
<tr>
<td>161</td>
<td>AM scale range over</td>
</tr>
<tr>
<td>162</td>
<td>not AM mode.</td>
</tr>
<tr>
<td>163</td>
<td>FM scale range over</td>
</tr>
<tr>
<td>164</td>
<td>not FM mode</td>
</tr>
<tr>
<td>165</td>
<td>Channel Bandwidth or Space range over in ACP</td>
</tr>
<tr>
<td>166</td>
<td>ACP error</td>
</tr>
<tr>
<td>167</td>
<td>Channel Bandwidth range over in Channel Power</td>
</tr>
<tr>
<td>168</td>
<td>Channel Power error</td>
</tr>
<tr>
<td>169</td>
<td>Reference Level Offset range over</td>
</tr>
<tr>
<td>999</td>
<td>Missing suffix and invalid suffix.</td>
</tr>
<tr>
<td>998</td>
<td>Unnecessary suffix insertion.</td>
</tr>
<tr>
<td>997</td>
<td>Bad command.</td>
</tr>
<tr>
<td>996</td>
<td>Input data size over error.</td>
</tr>
<tr>
<td>995</td>
<td>Processing the other function.</td>
</tr>
</tbody>
</table>