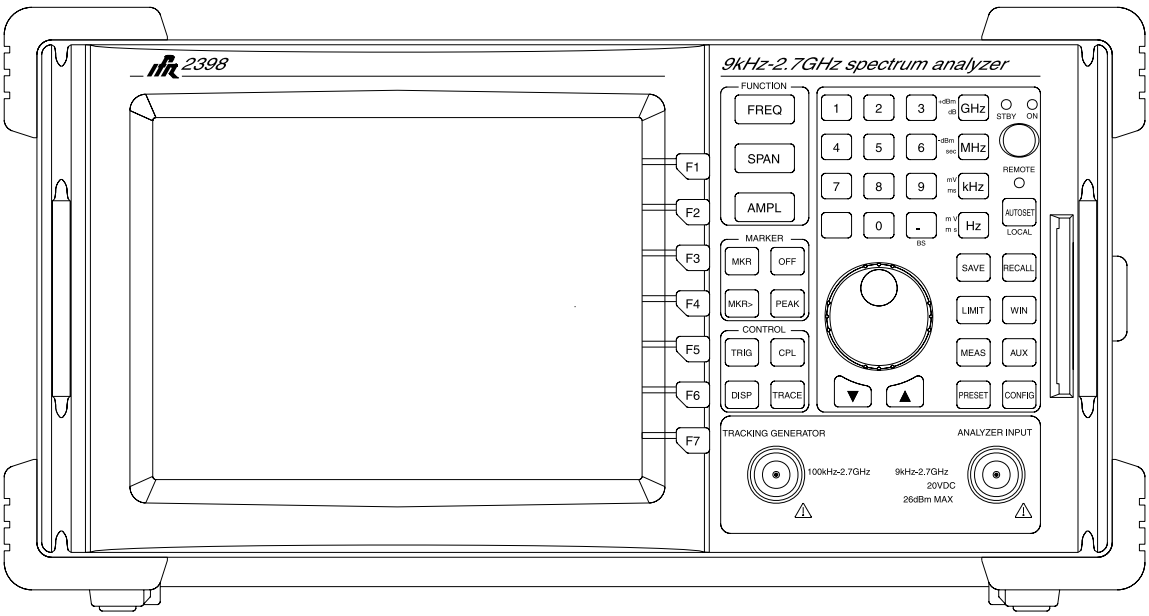
Spectrum Analyzer

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.*

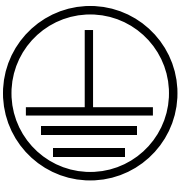
CAUTION. *Risk of hazard which caused fire or serious damage to the instrument or other equipment. Do not proceed until its suitable conditions are met.*

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.

Repair

WARNING ⚠

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.

Falling Over

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

CAUTION ⚠

Changing Fuse

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.

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.

Cleaning

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.

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 ⚠

Replacing Memory Back-Up Battery

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 7years. Early battery replacement is recommended.

CAUTION ⚠

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

Storage Medium

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

CAUTION ⚠

Product Damage Precaution

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.

Place-related Warninge

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 in 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.

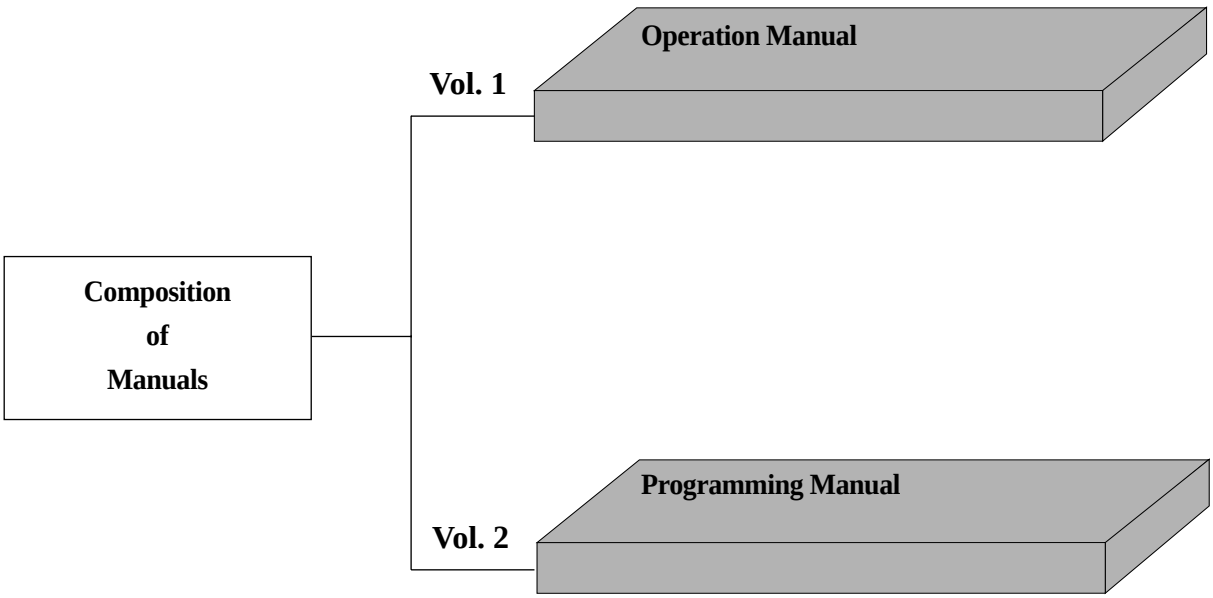
Measurement	Detector mode
◦ Normal signal	POS PEAK
◦ Random noise	SAMPLE OR AVERAGE
◦ Pulsed noise	NORMAL
◦ Occupied frequency bandwidth (for analog communication systems)	SAMPLE
◦ Occupied frequency bandwidth (for digital communication systems)	POS PEAK or SAMPLE

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

This section outlines the remote control and gives examples.

TABLE OF CONTENTS

〈BLANK〉

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

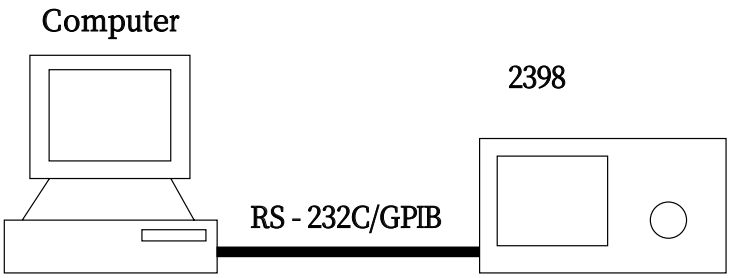
The 2398 Spectrum Analyzer has a standard RS - 232C interface, a GPIB interface bus and parallel(Printer) interface. Use the panel to select the interface port to be used to connect external devices as shown below.

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

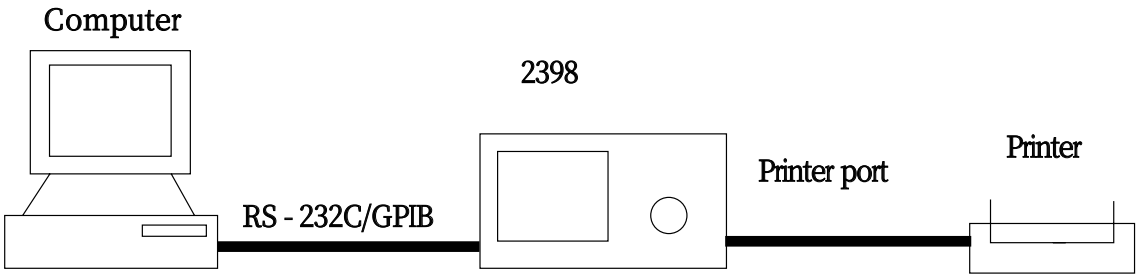
Examples of configurations using RS-232C and GPIB

(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.

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

Specifications of GPIB

The table below lists the specifications of the GPIB.

Item	Specification and supplementary explanation
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.

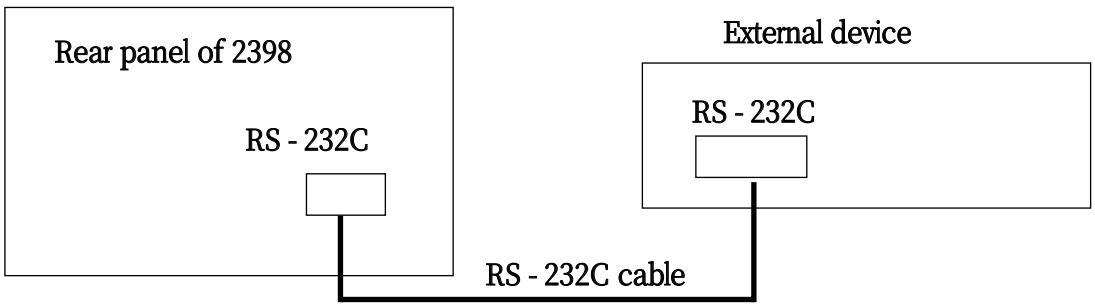
TABLE OF CONTENTS

〈BLANK〉

SECTION 2
CONNECTING DEVICES

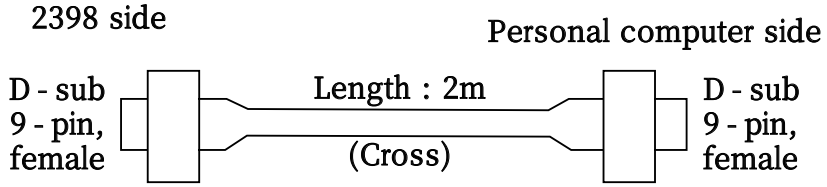
Connecting an external device with an RS-232C cable

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.



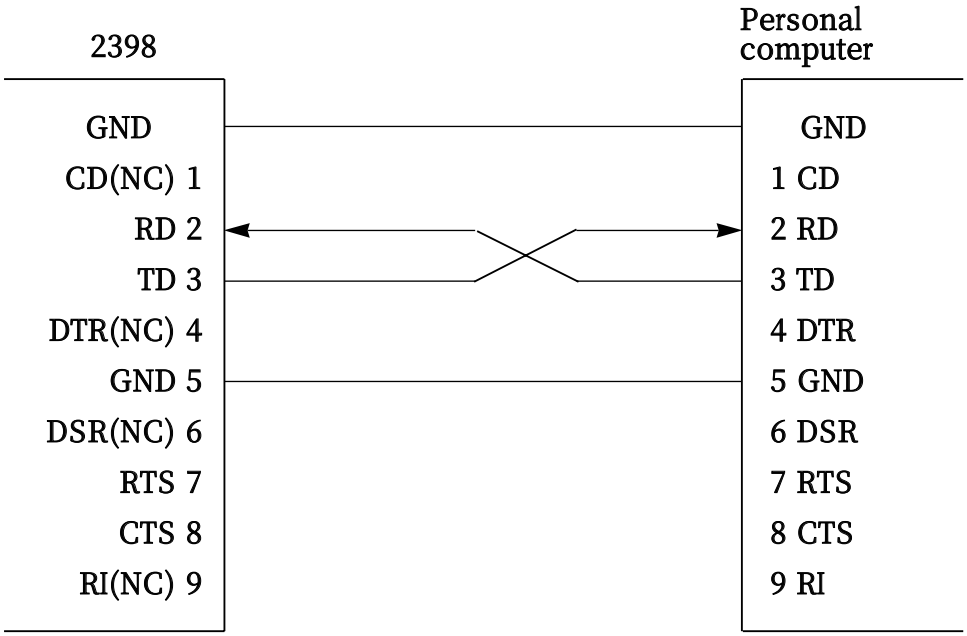
Notes : RS - 232C connectors with 9 pins are available, When purchasing the RS - 232C cable, check the pins on the RS - 232C connector of the external device.
Also, the following RS - 232C cables are provided as peripheral parts of the 2398.

● RS - 232C cable (for personal computer)



Connection diagram of RS-232C interface signals

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



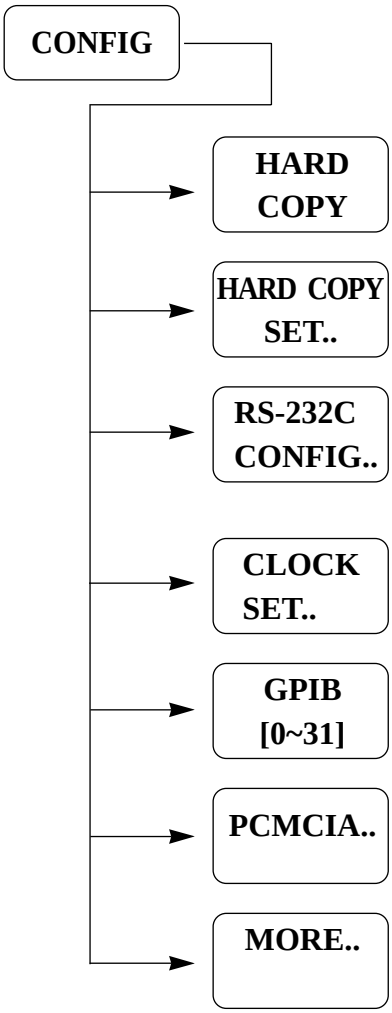
〈 Connection with personal computer 〉

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 turn 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.



Prints hard copy.

Selects HP Laserjet printer or PCMCIA CARD.

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

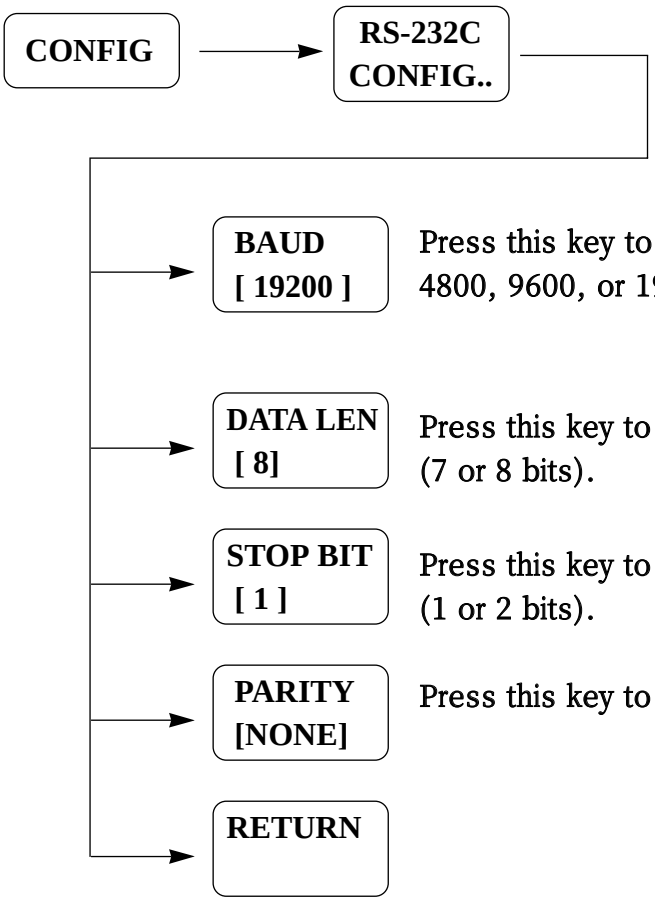
Selects the year, month, day, hour, minute, second.

Use the step or knob key to enter the GPIB address of this equipment.

PCMCIA check and format.

Setting the RS-232C interface conditions

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



Press this key to select a baud rate of 600, 1200, 2400, 4800, 9600, or 19200 bits per second.

Press this key to select the number of data bits (7 or 8 bits).

Press this key to select the number of stop bits (1 or 2 bits).

Press this key to select the parity (none, even, or odd)

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.

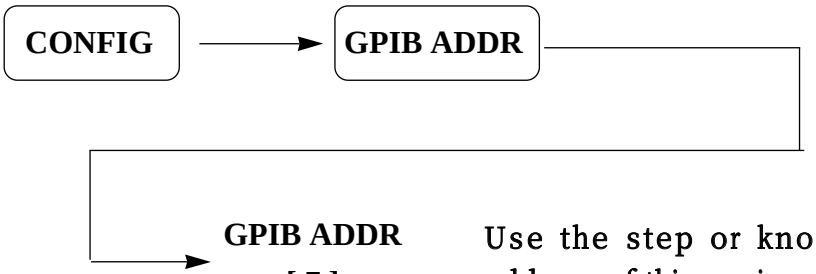
Note : Be sure to connect the GPIB cable before turning the equipment power on.

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.



GPIB ADDR
[7]

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

〈BLANK〉

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.

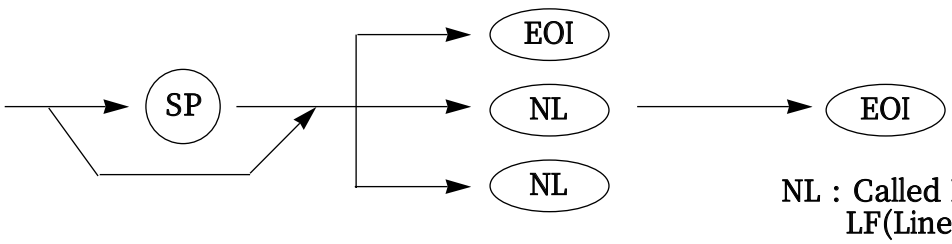
Program message format

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.

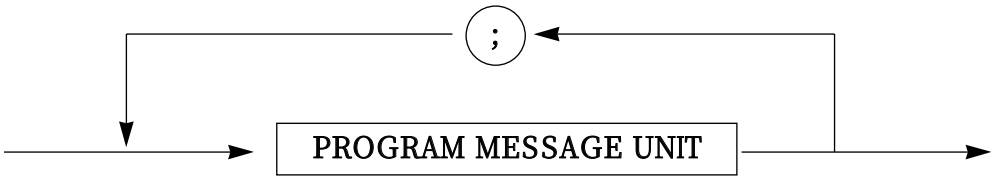
(1) PROGRAM MESSAGE TERMINATOR



NL : Called New Line or LF(Line Feed)

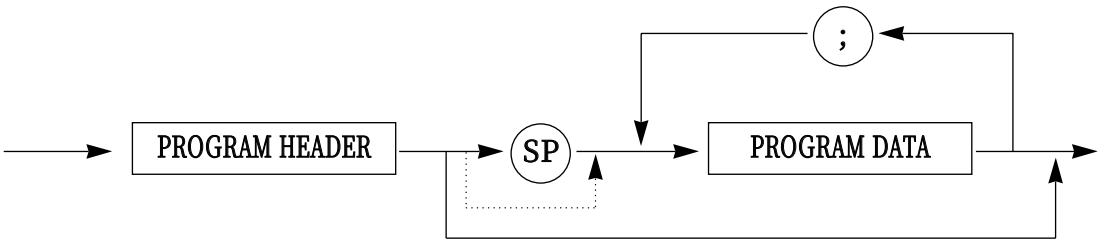
Carriage Return(CR) is ignored and is not processed as a terminator.

(2) PROGRAM MESSAGE



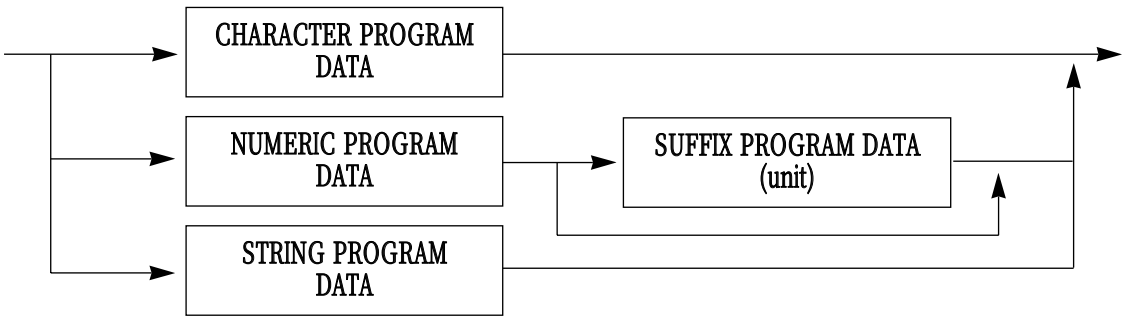
Multiple program message units can be output sequentially by separation them with a semicolon.
<Example> Send(" CF 1GHZ; SP 500MHZ; ");

(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

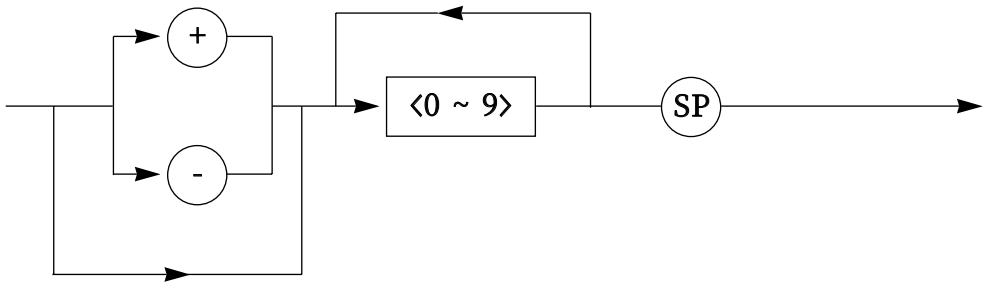
Character program data is specific character string data consisting of the upper - case alphabetic characters from A to Z, numbers 0 to 9, #, *, ?,

< Example > Send(" ST AUTO; "); ..... Sets Sweep Time to AUTO.

(6) NUMERIC PROGRAM DATA

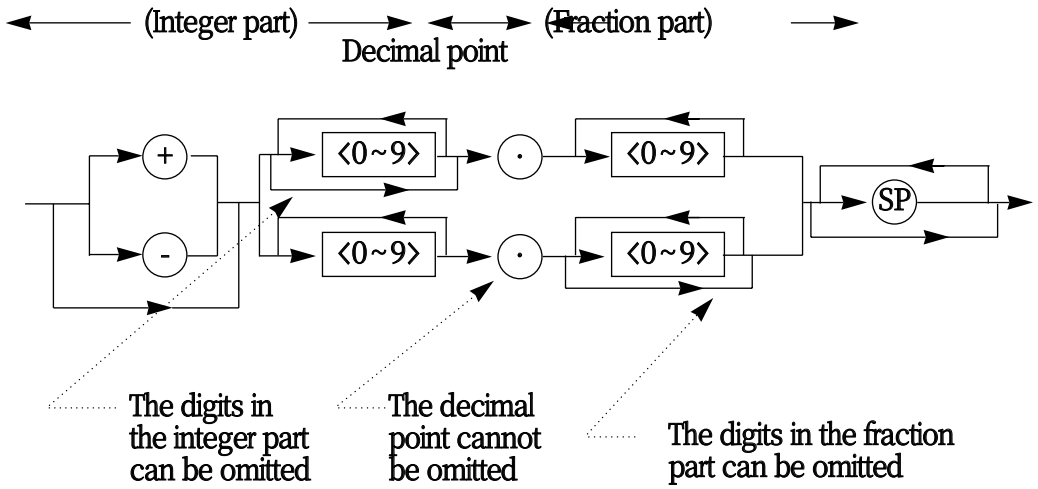
Numeric program data has two types of formats : integer format (NR1) and fixed - point format (NR2).

〈 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, +△5(×)
- Spaces can be inserted after a number → +5△△△
- The + sign is optional → +5.5
- Commas cannot be used to separate digits → 1,234,567 (×)

〈 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 → +753△.123(×)
- Spaces can be inserted after the digits in the fraction part → +753.123△△△
- 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.

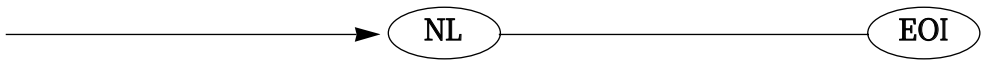
Table of 2398 Suffix Codes

Classification	Unit	Suffix code
Frequency	GHz	GHZ
	MHz	MHZ
	kHz	KHZ
	Hz	HZ
	Default	HZ
Time	Second	SEC
	m second	MS
	μsecond	US
	Default	MS
Level (dB system)	dB	DB
	dBm	DBM
	dBuV	DBUV
	dBmV	DBMV
	Default	Determined in conformance with the set scale unit
Level (V system)	V	V
	mV	MV
	uV	UV
	Default	Determined in conformance with the set scale unit
Level (W system)	W	W
	mW	MW
	uW	UW
	nW	NW
	pW	PW
	Default	Determined in conformance with the set scale unit

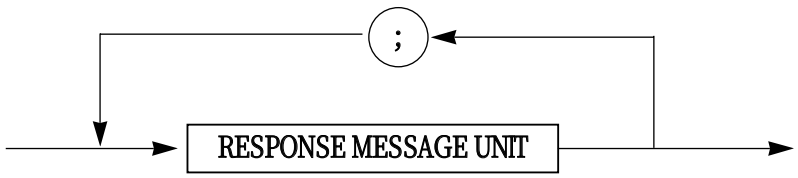
Response message format

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

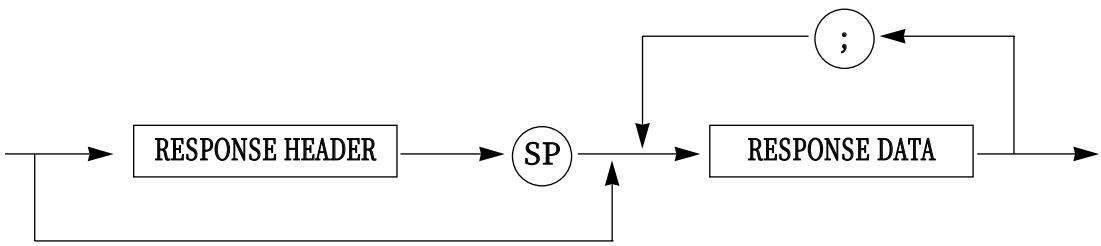


(2) RESPONSE MESSAGE

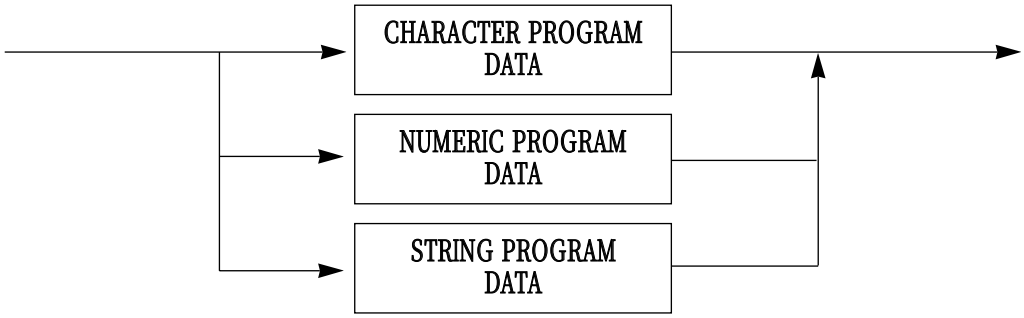


When a query is sent by the " Send " statement with one or more program queries, the response message also consists of one or more response message units.

(3) Usual RESPONSE MESSAGE UNIT



(4) RESPONSE DATA

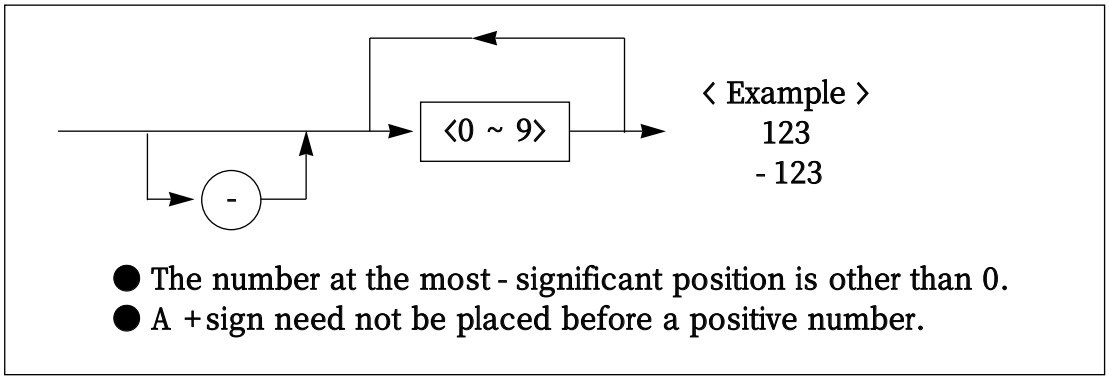


(5) CHARACTER RESPONSE DATA

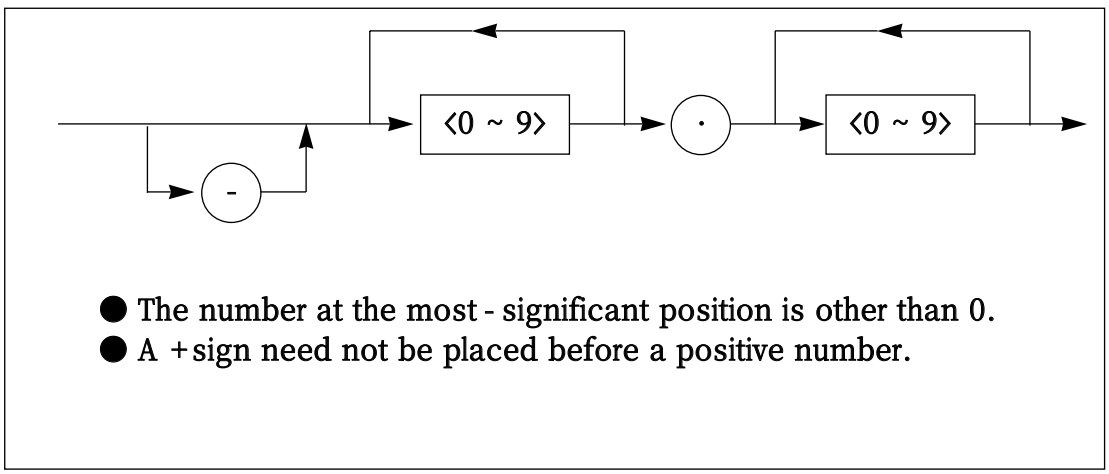
Character response data is specific character string data consisting of the upper - case alphabetic characters from A to Z, lower - case alphabetic characters from a to z, 0 to 9,and [,], dot(.), minus(-), comma(,).

(6) NUMERIC RESPONSE DATA

〈 Integer format (NR1) 〉



〈 Fixed - point format (NR2) 〉



SECTION 4
DETAILED DESCRIPTION OF COMMANDS

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

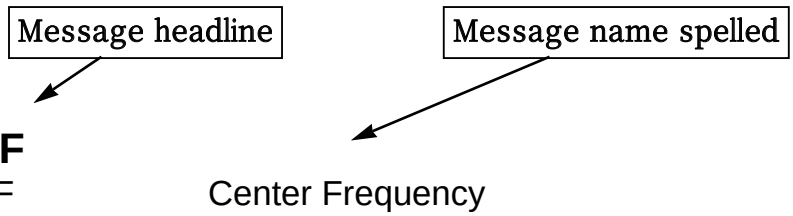
TABLE OF CONTENTS

<BLANK>

SECTION 4
DETAILED DESCRIPTION OF COMMANDS

General Description

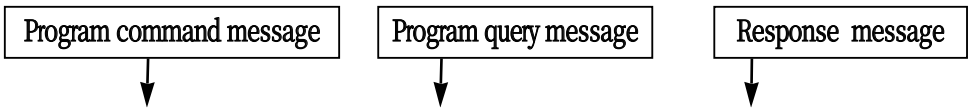
This section gives detailed descriptions of the device messages for the 2398 in functional order.



CF

CF Center Frequency

■ Function Sets the center frequency and sets the spectrum analyzer to center frequency/span mode.



Header	Program command	Query	Response
CF	CF△f CF△a	CF?	CF△f f = 0 to 2.7GHz Transfers the data with suffix code.

△ : Blank

Reference :" QRYTYP "command

■ 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 CF△123456;
CF△50MHZ;
CF?;

FREQUENCY

CF

CF Center Frequency

■ Function Sets the center frequency and sets the spectrum analyzer to center frequency / span mode.

Header	Program command	Query	Response
CF	CF△f CF△a	CF?	CF△f f = 0 to 2.7GHz Transfers the data with suffix code.

■ 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.

Header	Program command	Query	Response
FA	FA△f FA△a	FA?	FA△f f = 0 to 2.7GHz Transfers the data with suffix code.

■ 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 Stop Frequency

■ Function Sets the stop frequency and sets the spectrum analyzer to start frequency / stop frequency mode.

Header	Program command	Query	Response
FB	FB△f FB△a	FB?	FB△f f = 0 to 2.7GHz Transfers the data with suffix code.

■ 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.

Header	Program command	Query	Response
SS	SS△f SS△a	SS?	SS△f(AUTO), SS△f(MAN) f = 0 to 2.7GHz Transfers the data with suffix code.

■ 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

SP Frequency Span

■ Function Sets the frequency span.

Header	Program command	Query	Response
SP	SP△f SP△a	SP?	SP△f f = 0 to 2.7GHz Transfers the data with suffix code.

■ 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 × 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 Full Span

■ Function Selects the full frequency span.

Header	Program command	Query	Response
FS	FS	- - - -	- - - - -

■ Example FS;

ZS

ZS Zero Span

■ Function Sets zero frequency span.

Header	Program command	Query	Response
ZS	ZS	- - - -	- - - - -

■ Example ZS;

ZI

ZI Zoom-in

■ Function Changes to 1/2 the previous span.

Header	Program command	Query	Response
ZI	ZI	- - - -	- - - - -

■ Example ZI;

ZO

ZO Zoom-out

■ Function Changes to two times the previous span.

Header	Program command	Query	Response
ZO	ZO	- - - -	- - - - -

■ Example ZO;

AMPLITUDE

RL

RL Reference Level

■ Function Sets the reference level.

Header	Program command	Query	Response
RL	RL△l RL△a	RL?	RL△l l; units value depending on the current scale units.

■ 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;

AT

AT Input Attenuation

■ Function Sets the amount of attenuation for the input attenuator.

Header	Program command	Query	Response
AT	AT△a AT△n	AT?	AT△ndB(AUTO)/ AT△ndB(MAN) n= 0, 10, 20, 30, 40, 50

■ 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.

Header	Program command	Query	Response
LG	LG△l	LG?	LG△l l = 0,1,2,5,10 Zero(0) means not in LOG mode.

■ 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 Linear Mode

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

Header	Program command	Query	Response
LN	LN	LN?	LN△sw 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 Absolute Amplitude Units

■ Function Sets the absolute amplitude units for the input signal display.

Header	Program command	Query	Response
AUNITS	AUNITS△a	AUNITS?	AUNITS△u u = DBM, DBMV, DBUV, VOLT, 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

Header	Program command	Query	Response
RLO	RLO△l RLO△a	RLO?	RLO△res res = xx.xdB

■ 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 dB

■ Example RLO 6 dB;

MARKER

MKN

MKN Normal Marker

■ Function 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.

Header	Program command	Query	Response
MKN	MKN△f MKN△a MKN△t	MKN?	MKN△f f = 0 to 2.7GHZ MKN△t t = 0 to 20SEC (time resolution: sweep time/500)

■ 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;
```

MKA

MKA Marker Amplitude

■ Function Returns on the amplitude data in marker mode.

Header	Program command	Query	Response
MKA	- - - - - -	MKA?	MKA△l MKA△v MKA△w MKA△f MKA△p

■ Value of l 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).

Header	Program command	Query	Response
MKDTF	MKDTF	- - - -	- - - -

■ Example MKDTF;

MKD

MKD Marker Delta

■ Function Places delta marker on the normal marker position.

Header	Program command	Query	Response
MKD	MKD	- - - -	- - - -

■ Example MKD;

MKTF

MKTF Read the marker frequency or time

■ Function Returns time or frequency of a marker.

Header	Program command	Query	Response
MKTF	- - - -	MKTF?	MKTF△f(frequency) MKTF△t(time) Transfers the data with suffix code.

■ Example MKTF?;

MKOFF

MKOFF Marker Off

■ Function Turns off the marker mode.

Header	Program command	Query	Response
MKOFF	MKOFF	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKPK	MKPK△a	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKCF	MKCF	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKRL	MKRL	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKSP	MKSP	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKSS	MKSS	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKZI	MKZI	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKZO	MKZO	- - - -	- - - - -

■ 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.

Header	Program command	Query	Response
MKTRACK	MKTRACK△sw	MKTRACK?	MKTRACK△sw sw=0,1

■ 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.

Header	Program command	Query	Response
MKNOISE	MKNOISE△sw	MKNOISE?	MKNOISE△rsw rsw = OFF, Result Value

■ 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;

TRIG

TRGSWP

TRGSWP Trigger Sweep

■ Function Selects the continuous - sweep mode or the single - sweep mode.

Header	Program command	Query	Response
TRGSWP	TRGSWP△sw	TRGSWP?	TRGSWP△sw sw = 0, 1

■ 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.

Header	Program command	Query	Response
TM	TM△sw	TM?	TM△sw sw = FREE, VID, LINE, EXT

■ 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;

TF

TF Trigger Filter

■ Function Selects the trigger filter.

Header	Program command	Query	Response
TF	TF△sw	TF?	TF△sw sw = HPF, LPF

■ 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).

Header	Program command	Query	Response
TLV	TLV△x	TLV?	TLV△x x = 0 to 255

■ 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.

Header	Program command	Query	Response
TDLY	TDLY△t	TDLY?	TDLY△t t = - Sweep Time < t < Sweep Time Resolution : Sweep time / 500

■ 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?;

COUPLING

AUTOCPL

AUTOCPL Auto Coupled

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

Header	Program command	Query	Response
AUTOCPL	AUTOCPL	- - - -	- - - - -

■ Example AUTOCPL;

RB

RB Resolution Bandwidth

■ Function Sets the resolution bandwidth.

Header	Program command	Query	Response
RB	RB△a RB△f	RB?	RB△f(AUTO), RB△f(MAN) f = 300 to 3MHZ

■ 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.

Header	Program command	Query	Response
VB	VB△a VB△f	VB?	VB△f (AUTO), VB△f (MAN) f=10 to 1MHz, NONE

■ 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

ST Sweep Time

■ Function Sets the sweep time.

Header	Program command	Query	Response
ST	ST△a ST△t	ST?	ST△t(AUTO), ST△t(MAN) t=50ms to 1000S(sweep) t=5ms to 20S(zero span)

■ 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.

Header	Program command	Query	Response
DL	DL△sw DL△l	DL?	DL△rsw rsw = OFF, l (real)

■ 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)
W : W (MW : mW, UW : uW, NW : nW, PW : pW ,FW : fW)
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.

Header	Program command	Query	Response
TH	TH△sw TH△l	TH?	TH△rsw rsw = OFF , l(real)

■ 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)
W : W (MW : mW, UW : uW, NW : 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.

Header	Program command	Query	Response
TITLE	TITLE△text	TITLE?	TITLE△text text= ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789 (8 characters)

■ Value of text Character string within 8 characters.

■ Example TITLE IFR2398;
TITLE SPECTRUM;

CNTRST

CNTRST Contrast Control

■ Function Controls the display contrast.

Header	Program command	Query	Response
CNTRST	CNTRST△l	CNTRST?	CNTRST△l 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.

Header	Program command	Query	Response
GRAT	GRAT△sw	GRAT?	GRAT△sw sw=0,1

■ 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.

Header	Program command	Query	Response
ANNOT	ANNOT△sw	ANNOT?	ANNOT△sw sw=0,1

■ 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.

Header	Program command	Query	Response
CLEW	CLEW	- - - -	- - - -

■ 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.

Header	Program command	Query	Response
VIEW	VIEW	- - - - -	- - - - - - - - - - - -

■ Suffix code None

■ Example VIEW;

MXMH

MXMH Maximum Hold

■ Function Updates the chosen trace with the maximum signal level detected at each trace - data point from subsequent sweeps.

Header	Program command	Query	Response
MXMH	MXMH	- - - - - - -	- - - - - - - - - - - - -

■ Suffix code None

■ Example MXMH;

BLANK

BLANK Blank Trace

■ Function Blanks the trace from the display.

Header	Program command	Query	Response
BLANK	BLANK	- - - -	- - - -

■ Example BLANK;

TRA / TRB

TRA / TRB Trace Data Input / Output

■ 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).

Header	Program command	Query	Response
TRA	TRA△#ns△#ne,<b0 ···bn>	TRA△#ns△#ne,?	b = Binary data (2 bytes)
TRB	TRB△#ns△#ne,<d0,···dn>	TRB△#ns△#ne,?	[b1b2···bn]
	TRA△#ns△#ne,<b0···bn>		d = Decimal data (ASCII code)
	TRB△#ns△#ne,<d0,···dn>		[d1,d2,···dn]

■ 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).

Header	Program command	Query	Response
TRAALL TRBALL	- - - - - -	TRAALL? TRBALL?	b = Binary data (2 bytes) [b1b2···b1000] d = Decimal data (ASCII code) [d1,d2,···d500]

■ 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

■ Function 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.

Header	Program command	Query	Response
TDF	TDF△sw	TDF?	TDF△sw sw = BIN, DEC

■ 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

■ Function Sets the trace status.

Header	Program command	Query	Response
TRSTS	TRSTS△sw	TRSTS?	TRSTS A=rsw B=rsw rsw = 0, 1, 2, 3

■ 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 Trace A Minus Trace B

■ Function Subtracts the contents of Trace B from Trace A and places the result in Trace A.

Header	Program command	Query	Response
AMB	AMB	- - - - -	- - - - - - - -

■ Example AMB;

BML

BML Trace B Minus Display Line

■ Function Subtracts the display line from Trace B and places the result in Trace B.

Header	Program command	Query	Response
BML	BML	- - - - -	- - - - - - - - -

■ Example BML;

APB

APB Trace A Plus Trace B

■ Function Adds the contents of Trace B to Trace A and stores the result in Trace A.

Header	Program command	Query	Response
APB	APB	- - -	- - - -

■ Example APB;

AMBPL

AMBPL Trace A Minus Trace B Plus Display Line

■ Function Subtracts the contents of Trace B from Trace A, adds the display line to this value, and stores the result in Trace A.

Header	Program command	Query	Response
AMBPL	AMBPL	- - - - -	- - - - -

■ Example AMBPL;

AXB

AXB Trace A Exchange Trace B

■ Function Exchanges the contents of Trace A with those of Trace B.

Header	Program command	Query	Response
AXB	AXB	- - -	- - - -

■ Example AXB;

DETECT MODE

DET

DET Detection Mode

■ Function Selects the detection mode for the waveform data being displayed.

Header	Program command	Query	Response
DET	DET△d	DET?	DET△d 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;
```

AVERAGE

AVG

AVG Trace Average

■ Function Trace average on or off.

Header	Program command	Query	Response
AVG	AVG△sw	AVG?	AVG△sw sw = 0,1

■ Value of sw 1, ON : On
0, OFF : Off

■ Suffix code None

■ Initial setting OFF

■ Example AVG ON;

AVGC

AVGC Number of Trace Average

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

Header	Program command	Query	Response
AVGC	AVGC△n	AVGC?	AVGC△n n = 2 to 256

■ 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.

Header	Program command	Query	Response
AVGCYL	AVGCYL△sw	AVGCYL?	AVGCYL△sw sw=0,1

■ 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.

Header	Program command	Query	Response
AVGS	AVGS△sw	AVGS?	AVGS△sw sw = STOP, CONT

■ Value of sw STOP : Stop
CONT : Continue
RSET : Reset

■ Suffix code None

■ Initial setting CONT

■ Example AVGS CONT;

AUTOSET

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.

Header	Program command	Query	Response
AUTOSET	AUTOSET	- - - -	- - - -

■ Example AUTOSET;

SAVE

SVS

SVS Save State into Internal Register

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

Header	Program command	Query	Response
SVS	SVS△n	- - - -	- - - - - -

■ Value of n 0 to 9 (Memory number)

■ Suffix code None

■ Example SVS 1;

SVMS

SVMS Save State into Memory card

■ Function 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.

Header	Program command	Query	Response
SVMS	SVMS	- - - -	- - - - - -

■ Suffix code None

■ Example SVMS;

SVTRA

SVTRA Save Trace A into Internal Register

■ Function Saves Trace A in the specified trace register.

Header	Program command	Query	Response
SVTRA	SVTRA△n	- - - -	- - - - - -

■ 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.

Header	Program command	Query	Response
SVMTRA	SVMTRA	- - - -	- - - - - -

■ Suffix code None

■ Example SVMTRA;

SVTRB

SVTRB Save Trace B into Internal Register

■ Function Saves Trace B in the specified trace register.

Header	Program command	Query	Response
SVTRB	SVTRB△n	- - - -	- - - - - -

■ Value of n 0 to 9 (Memory number)

■ Suffix code None

■ Example SVTRB 1;

SVMTRB

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.

Header	Program command	Query	Response
SVMTRB	SVMTRB	- - - -	- - - - - -

■ Suffix code None

■ Example SVMTRB;

SVLMT

SVLMT Save Mask data into Internal Register

■ Function Saves the current-displayed mask data in the specified register.

Header	Program command	Query	Response
SVLMT	SVLMT△n	- - - -	- - - - - -

■ 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.

Header	Program command	Query	Response
SVMLMT	SVMLMT	- - - -	- - - - - -

■ Suffix code None

■ Example SVMLMT;

SVLCK

SVLCK Save Lock on or off

■ Function Do or don 't save in internal register.

Header	Program command	Query	Response
SVLCK	SVLCK△sw	SVLCK?	SVLCK△sw sw = 0, 1

■ 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.

Header	Program command	Query	Response
RCS	RCS△n	- - - -	- - - - - -

■ 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.

Header	Program command	Query	Response
RCTRA	RCTRA△n	- - - -	- - - - - -

■ 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.

Header	Program command	Query	Response
RCTRB	RCTRB△n	- - - -	- - - - - -

■ Value of n 0 to 9 (Memory number)

■ Suffix code None

■ Example RCTRB 1;

RCLMT

RCLMT Recall Mask data from Internal Register

■ Function Recalls previously - saved mask data to the display.

Header	Program command	Query	Response
RCLMT	RCLMT△n	- - - -	- - - - - -

■ 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.

Header	Program command	Query	Response
RCM	RCM△t.e	- - - -	- - - -

■ 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.

Header	Program command	Query	Response
LMTPC	LMTPC△sw	LMTPC?	LMTPC△rsw rsw= OFF, UFIL, LFIL, PASS, AFIL

■ Value of sw ON ; On
OFF ; Off

■ Value of rsw 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

Header	Program command	Query	Response
ALARM	ALARM△sw	ALARM?	ALARM△rsw rsw= ON, OFF

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

■ Suffix code None

■ Initial setting OFF ;off

■ Example ALARM ON;

LMTUP

LMTUP Limit Line Upper Area On / Off

■ Function Sets the upper limit line area on or off.

Header	Program command	Query	Response
LMTUP	LMTUP△sw	LMTUP?	LMTUP△sw sw =0, 1

■ Value of sw 1, ON ; On
0, OFF ; Off

■ Suffix code None

■ Initial setting OFF

■ Example LMTUP ON;

LMTLW

LMTLW Limit Line Lower Area On / Off

■ Function Sets the lower limit line area on or off.

Header	Program command	Query	Response
LMTLW	LMTLW△sw	LMTLW?	LMTLW△sw sw = 0, 1

■ 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.

Header	Program command	Query	Response
LMTUPD	LMTUPD△#1△#500,<d0,…dn>	LMTUPD△#1△#500,?	d = Decimal data (ASCII code)
LMTLWD	LMTLWD△#1△#500,<d0,…dn>	LMTLWD△#1△#500,?	[d0,d1,…dn]

■ 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,?;
```

WINDOW

WIN

WIN Window Function On / Off

■ Function Sets the Window function on or off.

Header	Program command	Query	Response
WIN	WIN△sw	WIN?	WIN△sw sw = 0,1

■ 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.

Header	Program command	Query	Response
WINUP	WINUP	- - - -	- - - - - - - -

■ Example WINUP;

WINLW

WINLW Lower Window

■ Function Selects the lower window area.

Header	Program command	Query	Response
WINLW	WINLW	- - - -	- - - - - - - -

■ Example WINLW;

WINUZ

WINUZ Window Upper Zoom-in

■ Function Changes the upper window area to the entire display.

Header	Program command	Query	Response
WINUZ	WINUZ	- - - -	- - - - -

■ Example WINUZ;

WINLZ

WINLZ Window lower Zoom-in

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

Header	Program command	Query	Response
WINLZ	WINLZ	- - - -	- - - - -

■ Example WINLZ;

WINT

WINT Window Toggle

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

Header	Program command	Query	Response
WINT	WINT△sw	WINT?	WINT△sw sw = 0, 1

■ 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.

Header	Program command	Query	Response
XDBDW	XDBDW△l	XDBDW?	XDBDW△l l = 3 to 80 (0.1dB step)

■ 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.

Header	Program command	Query	Response
XDBLW	XDBLW△l	XDBLW?	XDBLW△l l = 3 to 80 (0.1dB step)

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

XDBRW

XDBRW X dB Right Down

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

Header	Program command	Query	Response
XDBRW	XDBRW△l	XDBRW?	XDBRW△l l = 3 to 80 (0.1dB step)

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

XDBSEL

XDBSEL Select the X dB Marker display Format

■ Function Selects the X dB marker display format.

Header	Program command	Query	Response
XDBSEL	XDBSEL△sw	XDBSEL?	XDBSEL△sw sw = REL, ABSR, ABSL

■ Value of sw REL : Relative
ABSR : Right Absolute
ABSL : Left Absolute
■ Suffix code None
■ Initial setting REL
■ Example XDBSEL REL;

XDBSGL

XDBSGL Single Sweep and X dB Measurement

■ Function X dB down measurement is executed only once with sweep is finished.

Header	Program command	Query	Response
XDBSGL	XDBSGL	- - - - - -	- - - - - - - -

■ Example XDBSGL;

XDBCTN

XDBCTN Continuous Sweep and X dB Measurement.

■ Function X dB down measurement is executed continuously, after each sweep.

Header	Program command	Query	Response
XDBCTN	XDBCTN	- - - - -	- - - - -

■ Example XDBCTN;

XDBEND

XDBEND Stop the X dB Measurement function

■ Function Stops the X dB Measurements function.

Header	Program command	Query	Response
XDBEND	XDBEND	- - - -	- - - - - - -

■ Example XDBEND;

XDBRF

XDBRF Return Frequency result of the X dB Measurement

■ Function Returns the frequency result of the X dB Measurements.

Header	Program command	Query	Response
XDBRF	- - - - -	XDBRF?	XDBRF△f f = 0 to 2.7GHz

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

■ Example XDBRF?;

XDBRA

XDBRA Return Amplitude result of the X dB Measurement

■ Function Returns the amplitude result of the X dB Measurements.

Header	Program command	Query	Response
XDBRA	- - - - -	XDBRA?	XDBRA△l l = - 100dBm to +30dBm(or dB)

■ Value of l - 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.

Header	Program command	Query	Response
OBW	OBW△n	OBW?	OBW△n n = 10.0 to 99.8

■ 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 Return

■ Function Returns occupied power bandwidth measurements value.

Header	Program command	Query	Response
OBWR	- - - - - - -	OBWR?	OBWR△f f = 0 to 2.7GHz

■ 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

CHP Channel Power

■ Function Measuring channel power.

Header	Program command	Query	Response
CHP	CHP	CHP?	CHP△res res = xx.xx dBm

■ Suffix code None

■ Example CHP?;
CHP;

CHPHZ

CHPHZ Channel Power Spectral Density

■ Function Read channel power spectral density

Header	Program command	Query	Response
CHPHZ	- - - - -	CHPHZ?	CHPHZ△res res = xx.xx dBm/Hz

■ Suffix code None

■ Example CHPHZ?;

CHPCB

CHPCB Channel Bandwidth in Channel Power

■ Function Set channel bandwidth in channel power measurement.

Header	Program command	Query	Response
CHPCB	CHPCB△f CHPCB△a	CHPCB?	CHPCB△res res = 20Hz~1GHz

■ 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
```
CHPCB 2000;
CHPCB 1.23MHZ;
CHPCB?;
```

Adjacent Channel Power

ACP

ACP Adjacent Channel Power Ratio

■ Function Measuring adjacent channel power ratio.

Header	Program command	Query	Response
ACP	ACP	- - - - -	- - - - -

■ Suffix code None

■ Example ACP;

ACPC

ACPC Carrier Channel Power

■ Function Read carrier channel power in ACP

Header	Program command	Query	Response
ACPC	- - - - -	ACPC?	ACPC△res res = xx.xx dBm

■ Suffix code None

■ Example ACPC?;

ACPL

ACPL Lower Channel Power Ratio in ACP

■ Function Read lower channel power ratio in ACP measurement.

Header	Program command	Query	Response
ACPL	- - - - -	ACPL	ACPL△res res = xx.xx dB

■ Suffix code None

■ Example ACPL?;

ACPU

ACPU Upper Channel Power Ratio in ACP

■ Function Read upper channel power ratio in ACP measurement.

Header	Program command	Query	Response
ACPU	- - - - -	ACPU?	ACPU△res res = xx.xx dB

■ Suffix code None

■ Example ACPU?;

ACPCB

ACPCB Channel Bandwidth in ACP

■ Function Set channel bandwidth in ACP measurement.

Header	Program command	Query	Response
ACPCB	ACPCB△f ACPCB△a	ACPCB?	ACPCB△res res = 20Hz~1GHz

■ 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.

Header	Program command	Query	Response
ACPSP	ACPSP△f ACPSP△a	ACPSP?	ACPSP△res res = 20Hz~1GHz

■ 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 Quasi Peak Mode

■ Function Selects quasi peak mode bandwidth.

Header	Program command	Query	Response
QPM	QPM△sw	QPM?	QPM△sw sw = BNDB, BNDC, OFF

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

■ Suffix code None

■ Initial setting OFF

■ Example QPM BNDB;

Frequency Counter

MKFC

MKFC Frequency Counter

■ Function Activates a frequency counter that counts the frequency of the normal marker.

Header	Program command	Query	Response
MKFC	MKFC△sw	MKFC?	MKFC△sw sw= OFF, freq. freq. = 0 to 2.7GHz

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

■ Suffix code None

■ Initial setting OFF

■ Example MKFC ON;

MKFCR

MKFCR Frequency Counter Resolution

■ Function Specifies the resolution of the frequency counter.

Header	Program command	Query	Response
MKFCR	MKFCR△f	MKFCR?	MKFCR△f f=1, 10, 100, 1000 Transfer data with no unit of Hz

■ Value of f 1, 10, 100, 1000

■ Suffix code None

■ Initial setting 1000

■ Example MKFCR 1000;

AUXILIARY

DEMOD

DEMOD Demodulation

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

Header	Program command	Query	Response
DEMOD	DEMOD△sw	DEMOD?	DEMOD△sw sw = AM, FM, OFF

■ Value of sw AM : AM demodulation
FM : FM demodulation
OFF : Off

■ Suffix code NONE

■ Initial setting OFF

■ Example DEMOD AM;

AUDIO

AUDIO Speaker On or Off

■ Function Selects Speaker on or off.

Header	Program command	Query	Response
AUDIO	AUDIO△sw	AUDIO?	AUDIO△sw sw = 0, 1

■ Value of sw 1, ON : On
0, OFF : Off

■ Suffix code None

■ Initial setting 0, OFF

■ Example AUDIO ON;

SQL

SQL Squelch level

■ Function Adjusts squelch level.

Header	Program command	Query	Response
SQL	SQL△n	SQL?	SQL△n n = 0 to 255

■ Value of n 0 to 255 (1 step)

■ Suffix code None

■ Initial setting 115

■ Example SQL 115;

AUDIOVR

AUDIOVR Speaker Volume

■ Function Adjusts the volume of the speaker.

Header	Program command	Query	Response
AUDIOVR	AUDIOVR△n	AUDIOVR?	AUDIOVR△n n = 0 to 7

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

AMS

AMS AM Scale

■ Function Set AM scale

Header	Program command	Query	Response
AMS	AMS△sw	AMS?	AMS△rs rs = 25.0, 20.0, 10.0, 5.0, 2.5 0.0 = not FM mode

■ 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

Header	Program command	Query	Response
FMS	FMS△sw	FMS?	FMS△rs rs = 25.0, 20.0, 10.0, 5.0, 2.5, 1.0 0.0 = not FM mode

■ Value of sw 25, 20, 10, 5, 2.5, 1;
■ Suffix code None[kHz]
■ Initial setting 25
■ Example FMS 20;

PRESET

PRST

PRST Preset

■ Function Executes preset. All instrument parameters are set to default values.

Header	Program command	Query	Response
PRST	PRST	----------	----------

■ Example PRST;

CALALL

CALALL All Calibrations

■ Function Executes all calibrations.

Header	Program command	Query	Response
CALALL	CALALL	----------	----------

■ Example CALALL;

PCAL

PCAL Temperature Calibrations Executions On or Off

■ Function Initiates periodic temperature calibration execution.

Header	Program command	Query	Response
PCAL	PCAL△sw	PCAL?	PCAL△sw sw = 0, 1

■ Value of sw 1, ON : On
0, OFF : Off

■ Suffix code None

■ Initial setting ON

■ Example PCAL ON;

YIGCAL

YIGCAL First Local Oscillator Calibration

■ Function Executes First local oscillator Calibration.

Header	Program command	Query	Response
YIGCAL	YIGCAL	----------	----------

■ Example YIGCAL;

RCAL

RCAL RBW Calibration

■ Function Executes RBW Calibration.

Header	Program command	Query	Response
RCAL	RCAL	----------	----------

■ Example RCAL;

TMPCAL

TMPCAL Temperature Calibration

■ Function Executes temperature Calibration.

Header	Program command	Query	Response
TMPCAL	TMPCAL	----------	----------

■ Example TMPCAL;

LVLC

LVLC Level Calibration

■ Function Executes level calibration.

Header	Program command	Query	Response
LVLC	LVLC	----------	----------

■ Example LVLC;

SPCAL

SPCAL Span Calibration

■ Function Executes span calibration.

Header	Program command	Query	Response
SPCAL	SPCAL	----------	----------

■ Example SPCAL;

LAC

LAC LOG Calibration

■ Function Executes LOG amplifier calibration.

Header	Program command	Query	Response
LAC	LAC	----------	----------

■ Example LAC;

CALSIG

CALSIG Calibration Signal On or Off

■ Function Turns the calibration signal on or off.

Header	Program command	Query	Response
CALSIG	CALSIG△sw	CALSIG?	CALSIG△sw sw = 0, 1

■ Value of sw 1, ON : On
0, OFF : Off

■ Suffix code None

■ Initial setting ON

■ Example CALSIG ON;

ACAL

ACAL Amplitude Calibration

■ Function Turns amplitude calibration tables on or off.

Header	Program command	Query	Response
ACAL	ACAL△sw	ACAL?	ACAL△sw sw = ON, OFF

■ 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.

Header	Program command	Query	Response
HCOPY	HCOPY	- - - - -	- - - - - - - - - - - - - - - - - -

■ Example HCOPY;

HCPTYP

HCPTYP Hard Copy Type

■ Function Sets hard copy type.

Header	Program command	Query	Response
HCPTYP	HCPTYP△sw	HCPTYP?	HCPTYP△sw sw = PRT, CRD

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

■ Example HCPTYP PRT;

CLOCK SET

DATE

DATE Date

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

Header	Program command	Query	Response
DATE	DATE△yy.mm.dd	DATE?	DATE△yy.mm.dd yy.mm.dd

■ Value of yy 00 to 99 (year), 96~99:1996~1999, 00~95:2000~2095
■ Value of mm 01 to 12 (month)
■ Value of dd 01 to 31 (day)
■ Suffix code None
■ Example DATE 98.08.21;

TIME

TIME Time

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

Header	Program command	Query	Response
TIME	TIME△hh.mm.ss	TIME?	TIME△hh:mm:ss hh:mm:ss

■ 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.

Header	Program command	Query	Response
PCMCHK	- - - - -	PCMCHK?	PCMCHK△sw sw = -2, -1, 0, 1, 2

■ 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.

Header	Program command	Query	Response
REFLO	REFLO△sw	REFLO?	REFLO△sw sw = INT, EXT

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

■ Suffix code None

■ Initial setting INT

■ Example REFLO INT;

GPIB COMMON COMMAND

*CLS

*CLS Clear Status Command

■ Function Clears the status byte register.

Header	Program command	Query	Response
*CLS	*CLS	- - - -	- - - - - - - - -

■ Example *CLS;

*ESE

*ESE Standard Event Status Enable

■ Function Sets or clears the standard status enable register.

Header	Program command	Query	Response
*ESE	*ESE△n	*ESE?	*ESE△n n = 0 to 255

■ 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.

Header	Program command	Query	Response
*ESR	- - - - - - -	*ESR?	*ESR△n n = 0 to 255

■ Value of n 0 to 255

■ Suffix code None

■ Example *ESR?;

*IDN?

*IDN? Identification Query

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

Header	Program command	Query	Response
*IDN	- - - - - - -	*IDN?	IFR, 2398, serial, version

■ Suffix code None

■ Example *IDN?;

*OPC

*OPC Operation Complete Command

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

Header	Program command	Query	Response
*OPC	*OPC	- - - - - -	- - - - - - - - - - -

■ Example *OPC;

*OPC?

*OPC? Operation Complete Query

■ Function Sets the output queue to 1 to generate a MAV summary message when all pending select device operations have been completed.

Header	Program command	Query	Response
*OPC?	- - - - - - -	*OPC?	1

■ Example *OPC?;

*RST

*RST Reset Command

■ Function Resets the device.

Header	Program command	Query	Response
*RST	*RST	- - - - -	- - - - - -

■ Example *RST;

*SRE

*SRE Service Request Enable Command

■ Function Sets the bits in the service request enable register.

Header	Program command	Query	Response
*SRE	*SRE△n	*SRE?	*SRE△n n = 0 to 255

■ Value of n 0 to 255

■ Suffix code None

■ Example *SRE 1;
*SRE?;

*STB?

*STB? Returns Status Byte Command

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

Header	Program command	Query	Response
*STB	- - - - - - -	*STB?	*STB△n n = 0 to 255

■ Value of n

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

■ Example *STB?;

OTHERS

ESE2

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.

Header	Program command	Query	Response
ESE2	ESE2△n	ESE2?	ESE2△n n = 0 to 255

■ Value of n 0 to 255 : Represents the sum of the bit - weighted values.

■ Suffix code None

■ Example ESE2 1;
ESE2?;

ESR2?

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.

Header	Program command	Query	Response
ESR2?	- - - - - - -	ESR2?	ESR2△n n = 0 to 255

■ Value of n 0 to 255

■ Suffix code None

■ Example ESR2?;

QRYTYP

QRYTYP Query Response Type

■ Function Sets query response type.

Header	Program command	Query	Response
QRYTYP	QRYTYP△sw	QRYTYP?	QRYTYP△sw sw = ECO, NEC

■ 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

Header	Program command	Query	Response
ERR	- - - - - - -	ERR?	ERR△n n = xxx(x : 0 to 9)

■ 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

Header	Program command	Query	Response
ECHO	ECHO△sw	ECHO?	ECHO△res res = 1, 0

■ 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

Header	Program command	Query	Response
TGEN	TGEN△sw	TGEN?	TGEN△rsw rsw = 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

Header	Program command	Query	Response
TGLEV	TGLEV△l TGLEV△a	TGLEV?	TGLEV△l l = -70.0 0.0dBm 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 = 0.0dBm

■ Example TGLEV -12.5DBM;
TGLEV 0DBM;

TGNORM

TGNORM Tracking Generator Normalize

■ Function Sets the normalize function of tracking generator

Header	Program command	Query	Response
TGNORM	TGNORM△sw	TGNORM?	TGNORM△rsw rsw = ON, OFF

■ 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

Header	Program command	Query	Response
TGAFC	TGAFC	- - - -	- - - - - - -

■ Example TGAFC;

TGMFC

TGMFC Tracking Generator Manual Frequency Calibration

■ Function Executes frequency calibration for tracking generator manually

Header	Program command	Query	Response
TGMFC	TGMFC△f TGMFC△a	TGMFC?	TGMFC△f f = offset frequency

■ 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 OF CONTENTS

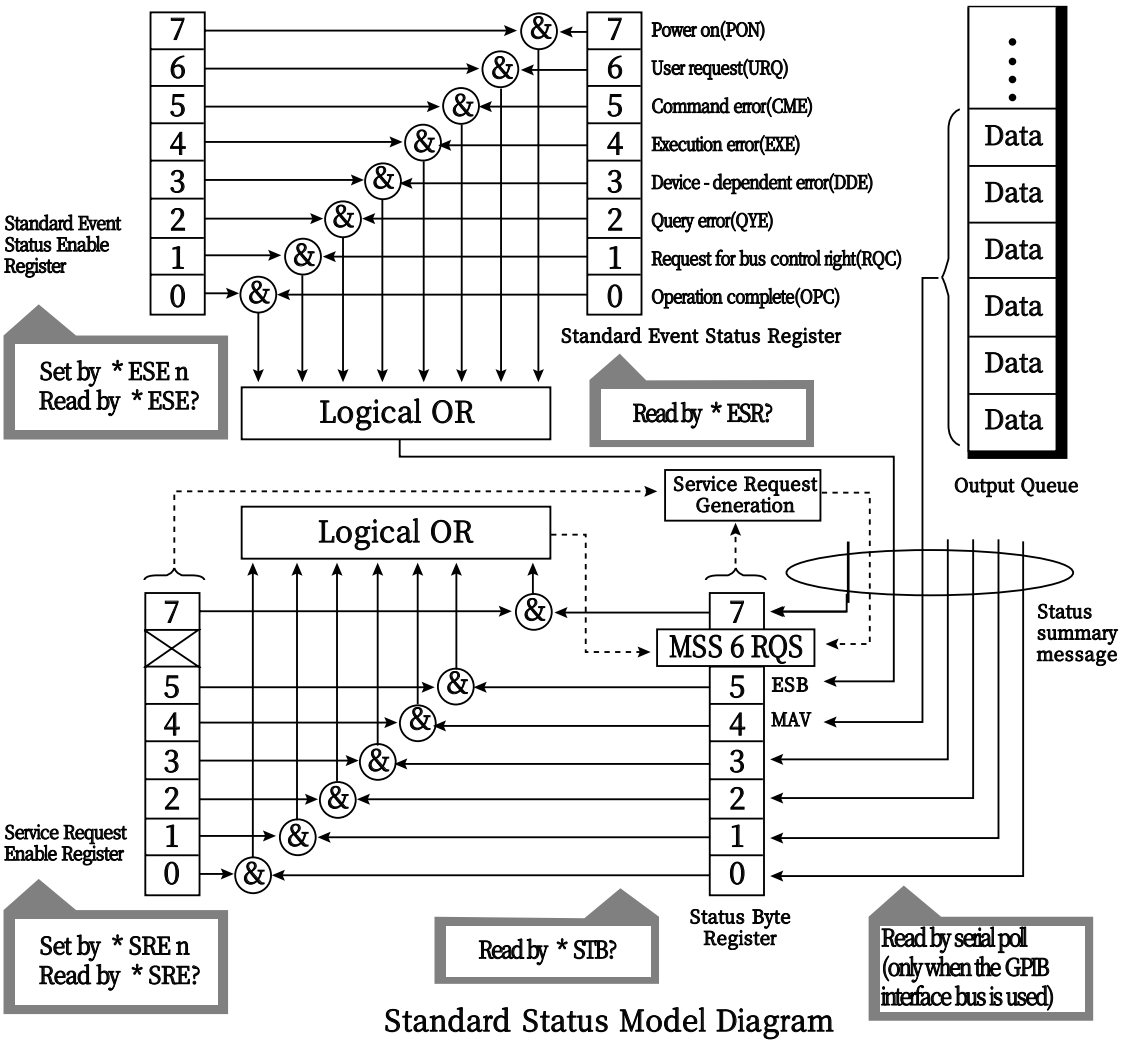
<BLANK>

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 grad status structure. To create these summary message bits, the status data structure is composed of two types of register and models.

Register model	Queue model
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.	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.

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

Standard Event Status Register	Status Byte Register	Output Queue
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 at 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).	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.	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.

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.

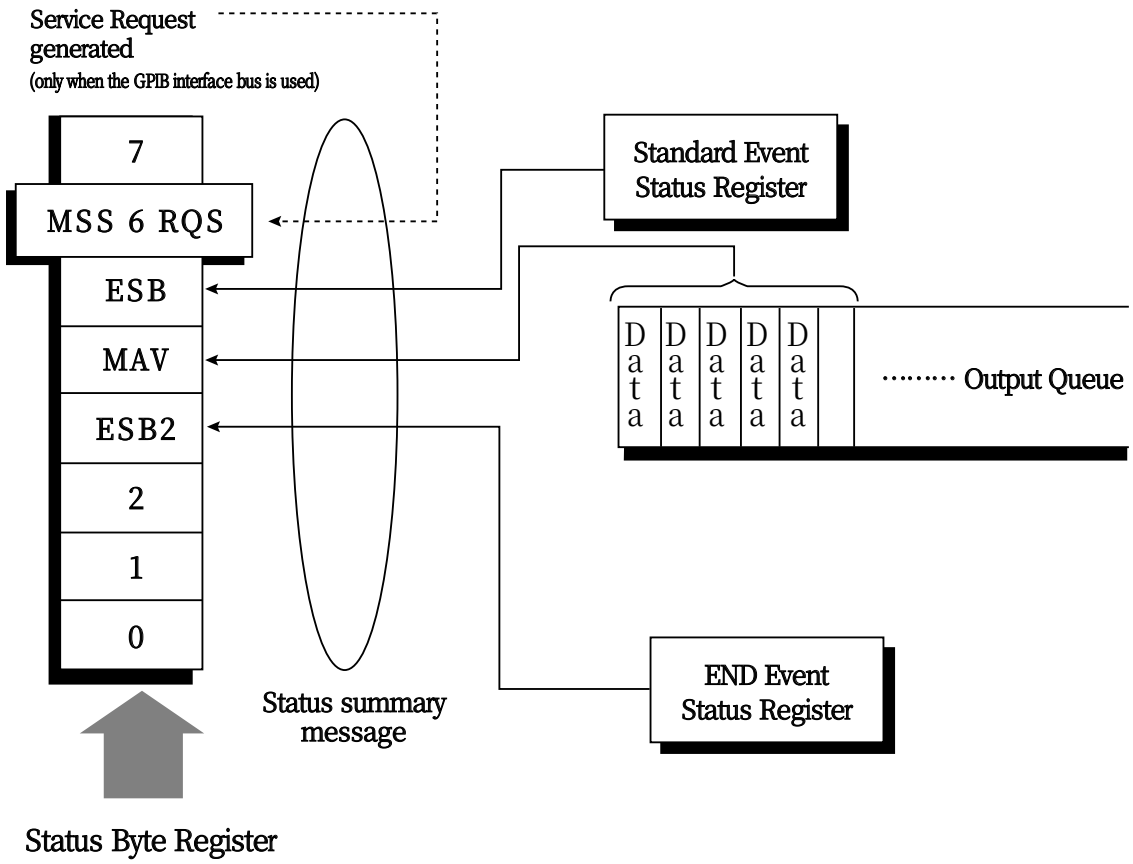
This bit becomes 0 when the ESR register is read by the *ESR? query or when it is cleared by the *CLS command.

(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.



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)

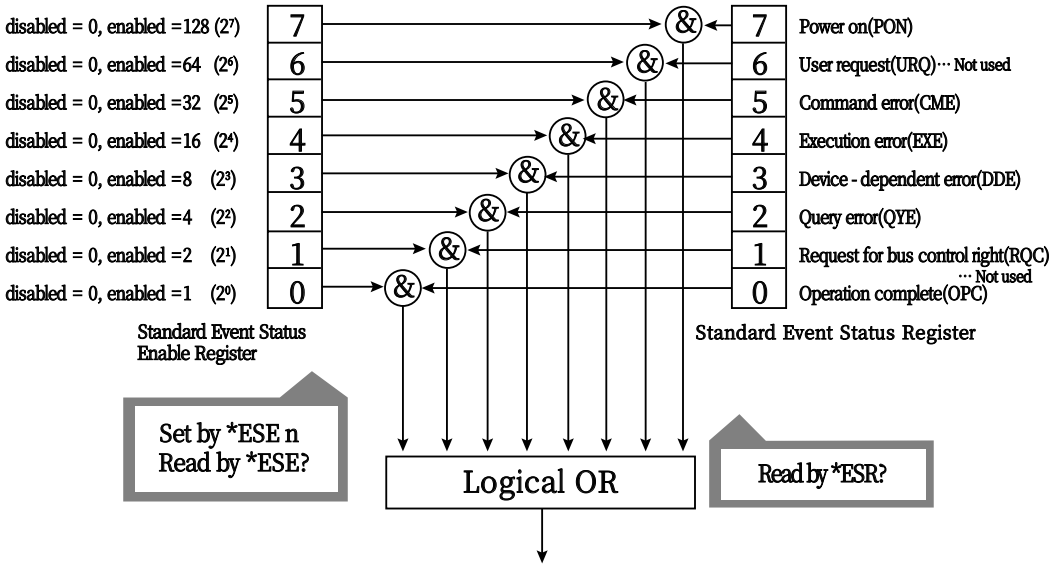
MSS indicates that there is at least one cause for a service request. The MSS message is represented at bit 6 response to an *STB? Query, but it is not produced as a response to serial polling. It should not be taken as part of the status byte specified by IEEE488.1.
MSS is configured by the over all 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.



(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.

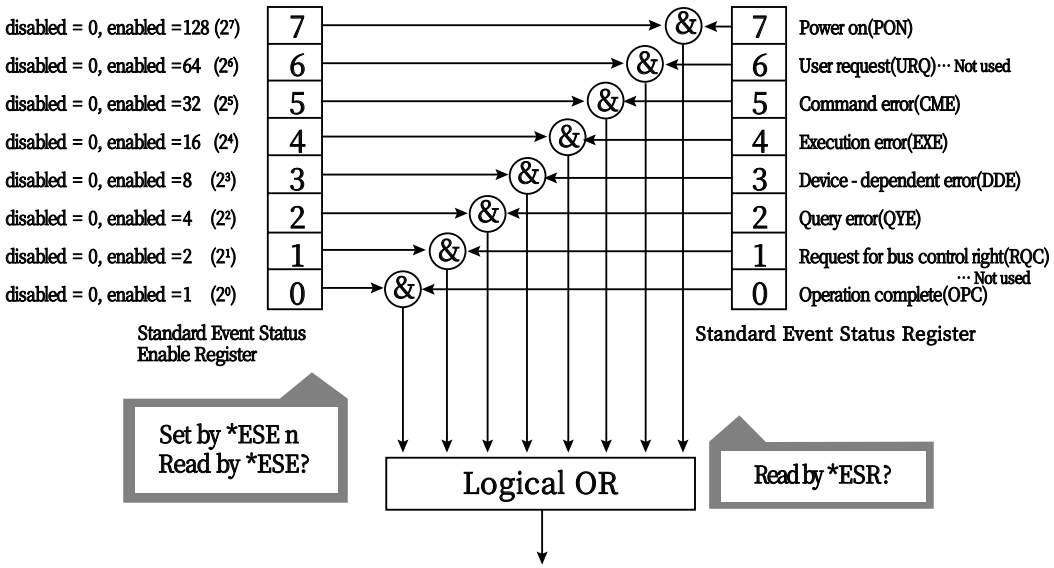
(2)Updating the SRE register

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.



ESB summary message bit
(to Status Byte register bit 5)

The Standard Event Status Enable (ESE) Register on the left is used to select which bits in the corresponding Event Register will cause a TRUE summary message when set.

Bit	Event name	Description
7	Power on (PON - Power on)	A transition from power - off to power - on occurred during the power - up procedure.
6	Not used	
5	Command error (CME - Command Error)	An illegal program message or a misspelled command was received.
4	Execution error (EXE - Execution Error)	A legal but unexecutable program message was received
3	Device - dependent error (DDE - Device - dependent Error)	An error not caused by CME, EXE, or QYE occurred (parameter error, etc.).
2	Query error (QYE - Query Error)	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
1	Not used	
0	Operation complete (OPC - Operation Complete)	This bit becomes 1 when this instrument has Processed the *OPC command.

Reading, writing, and clearing the Standard Event Status Register

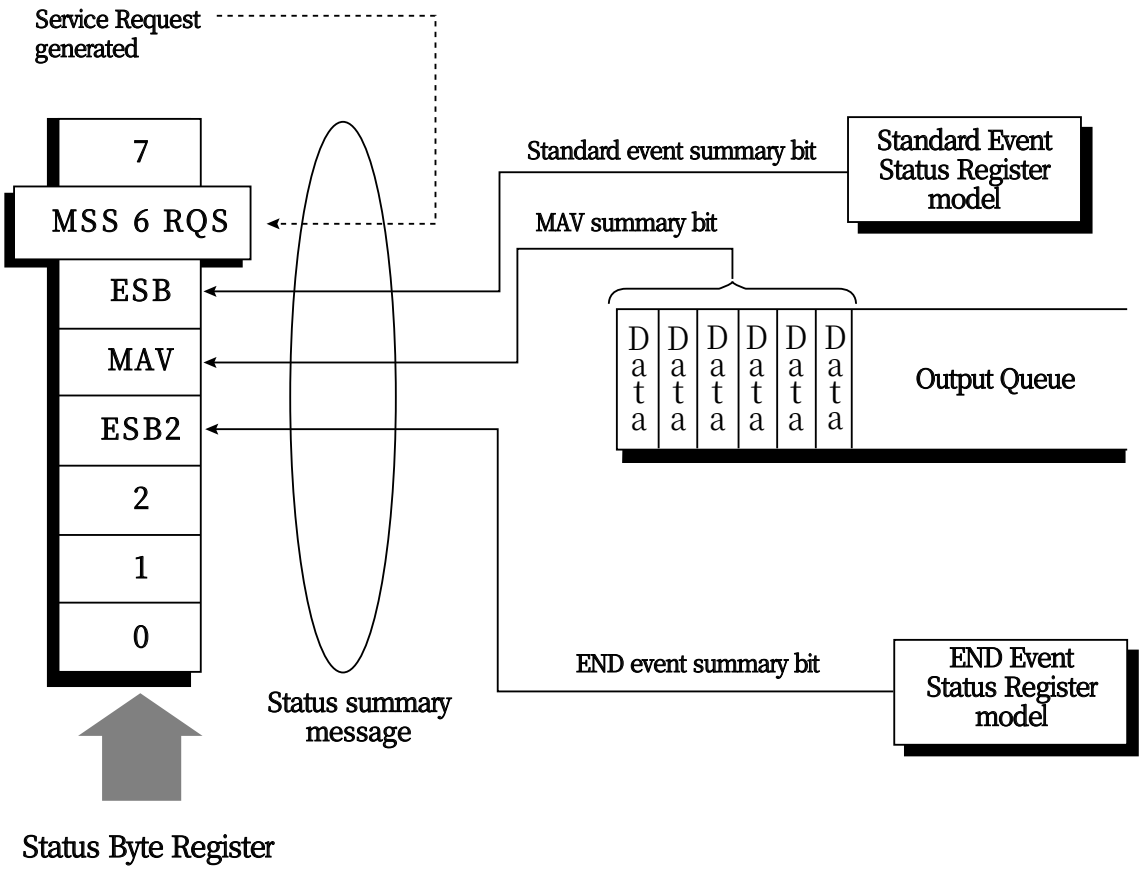
Reading	The register is read using the *ESR? command query. The register is cleared after being read. The response message is integerformat data with the binary weight added to the event bit and the sum converted to decimal.
Writing	With the exception of clearing, data cannot be written to the register from outside.
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

Reading	The registers 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.
Writing	The register is written using the *ESE common command.
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.

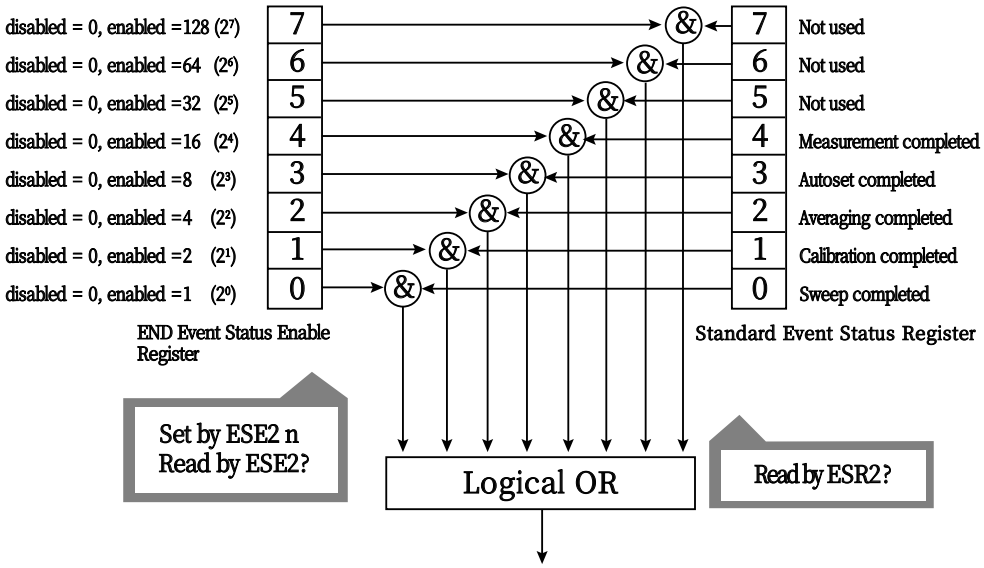
Extended Event Status Register

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.



The END Event Status Enable Register on the left is used to select which bits in the corresponding Event Register will cause a TRUE summary message when set.

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

Reading, writing, and clearing the Extended Event Status Register

Reading	The ESR2? common 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 to decimal.
Writing	With the exception of clearing, data cannot be written to the register from outside.
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 ESR2? query command.

Reading, writing, and clearing the Extended Status Enable Register

Reading	The ESE2? 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.
Writing	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.
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. The Extended Event Status 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.

〈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

〈BLANK〉

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

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

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

   // read data.
  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 initalization, 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("AUTOCPL;"); // 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("MKN?;");  // 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 SpollByte;

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

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

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

  // read data.
  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 initalization, 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("<<<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("MKA?;"); // Read the amplitude difference between the Delta and the Normal marker
 Receive();
 printf("Delta Amplitude = %s\n",DataBuf);

 Send("MKTF?;"); // 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 6dB 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)
 {
  ibwrt(ud, buf, (long)strlen(buf));
  if (ibsta & ERR) printf("ibwrt Error\n");
 }

 void Receive(void)
 {
  // read data.
  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 initalization, 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("XDBDW 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
   f. Unit: dBm, Log 10 dB scale,
   g. Reference Level: -25 dBm
   h. RBW: 10 kHz
   i. VBW: 1 kHz
   j. 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 "Decl-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 2398
  ud = ibdev(0,7,0,T10s,1,0); // GPIB initalization, 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)
 {
   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 2398
   ud = ibdev(0,7,0,T10s,1,0); // GPIB initalization, 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("<<<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 DBM;");    // 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.

```
//-------------------------------------------------------------------------
//            Save the current status to PCMCIA[OPTION]
//-------------------------------------------------------------------------
 #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)
 {
  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 2398
  ud = ibdev(0,7,0,T10s,1,0); // GPIB initalization, 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("<<<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(atoi(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.

```
//-------------------------------------------------------------------------
//            Recall the current status from PCMCIA[OPTION]
//-------------------------------------------------------------------------
 #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)
 {
   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 2398
   ud = ibdev(0,7,0,T10s,1,0); // GPIB initalization, 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.

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//                    Get all the Trace Data
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 #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)
 {
   ibwrt(ud, buf, (long)strlen(buf));
   if (ibsta & ERR) printf("ibwrt Error\n");
 }

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

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

   // read data.
   ibrd(ud, DataBuf, 4096L);
   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 initalization, 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)
{
    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[n++];
         TempBuf[j++]= c;
      }while( isdigit(c) );

      TempBuf[j] = '\0';
      TraceData[i] = atoi(TempBuf);
      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.

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

  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 2398
    ud = ibdev(0,7,0,T10s,1,0); // GPIB initalization, 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("<<<Pass / Fail Check>>>\n");

   InitGPIB();

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

   // Already, the Limit mask data had to be saved in internal register 3.
   // Recall the limit mask data from internal register 3.
   // When RCLMT 3 is completed, The configration is replaced by the data to have saved.
   Send("RCLMT 3;"); // load the limit mask data from internal memory 3.

   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);

}// the end of main
```

〈BLANK〉

APPENDIX - PROGRAMMING COMMANDS

CHP	Channel Power	4 - 64
CHPCB	Channel Bandwidth in Channel Power	4 - 65
CHPHZ	Channel Power Spectral Density	4 - 64
CLEW	Sets the chosen trace to clear - write mode.	4 - 33
*CLS	Clears the status byte register.	4 - 83
CNTRST	Controls the display contrast.	4 - 31
DATE	Sets the built - in clock to the specified date.	4 - 80
DEMOD	Activates either AM or FM demodulation.	4 - 72
DET	Selects the detection mode.	4 - 41
DL	Sets the display line.	4 - 29
ECHO	RS - 232C echo	4 - 89
ERR	Error number	4 - 89
*ESE	Sets or clears the standard status enable register.	4 - 83
ESE2	Event Status Enable (End)	4 - 88
*ESR?	Return the value in the standard event status register.	4 - 84
ESR2?	Event Status Register (End) Query	4 - 88
FA	Sets the start frequency	4 - 5
FB	Sets the stop frequency	4 - 6
FMS	Set FM scale	4 - 73
FS	Selects full frequency span	4 - 9
GRAT	Turns the display graticule on or off.	4 - 32
HCOPY	Prints hard copy.	4 - 79
HCPTYP	Sets hard copy type.	4 - 79
*IDN?	Return the model, software version NO. of the equipment.	4 - 84
LAC	Executes LOG calibrations.	4 - 77
LG	Selects logarithmic amplitude mode & dB scale.	4 - 12
LMTLW	Sets the lower limit line area on or off.	4 - 54
LMTPC	Sets the limit line function on or off.	4 - 53
LMTUP	Sets the upper limit line area on or off.	4 - 54

LMTUPD/LMTLWD	Upper / lower limit line data input / output.	4 - 55
LN	Selects a linear amplitude mode.	4 - 13
LVLC	Executes level calibration.	4 - 77
MKA	Returns the amplitude of the active maker.	4 - 16
MKCF	Marker to center frequency.	4 - 18
MKD	Places an delta marker on the normal marker position.	4 - 17
MKDTF	Marker 1/delta	4 - 16
MKFC	Activates frequency counter for the normal marker.	4 - 71
MKFCR	Specifies the resolution of the frequency counter.	4 - 71
MKN	Places an active marker on the specified frequency.	4 - 15
MKNOISE	Marker noise.	4 - 22
MKOFF	Turns off the marker mode.	4 - 17
MKPK	Peak Search.	4 - 18
MKRL	Marker to Reference level.	4 - 19
MKSP	Marker delta to span.	4 - 19
MKSS	Marker to center frequency step size.	4 - 20
MKTF	Read the marker frequency or time	4 - 17
MKTRACK	Signal track.	4 - 21
MKZI	Marker zoom - in.	4 - 20
MKZO	Marker zoom - out.	4 - 21
MXMH	Maximum Hold.	4 - 34
OBW	Calculates and displays the occupied power bandwidth.	4 - 63
OBWR	Returns occupied power bandwidth measurements value.	4 - 63
*OPC	Operation complete command	4 - 85
*OPC?	Operation complete query	4 - 85
PCAL	Sets temperature calibration execution.	4 - 75
PCMCHK	Returns the PCMCIA memory card status.	4 - 81
PRST	Executes preset.	4 - 74
QPM	Selects quasi peak mode bandwidth.	4 - 70
QRYTYP	Sets query response type.	4 - 89
RB	Sets the resolution bandwidth.	4 - 26

RCAL	Executes RBW calibration.	4 - 76
RCLMT	Recalls previously - saved mask data to the display.	4 - 51
RCM	Recalls the date from a memory card.	4 - 52
RCS	Recalls to the display a saved instrument state.	4 - 50
RCTRA	Recalls trace A from internal register.	4 - 50
RCTRB	Recalls trace B from internal register.	4 - 51
REFLO	Selects the reference clock.	4 - 82
RL	Sets the reference level.	4 - 11
RLO	Reference level offset	4 - 14
*RST	Resets the device.	4 - 86
SP	Sets the frequency span	4 - 8
SPCAL	Executes span calibration.	4 - 77
SQL	Adjusts squelch level.	4 - 72
*SRE	Sets the bits in the service request enable register.	4 - 86
SS	Sets the center frequency step size	4 - 7
ST	Sets the sweep time.	4 - 28
*STB?	Returns the status bytes including the MSS bit.	4 - 87
SVLCK	Save lock on or off.	4 - 49
SVLMT	Saves the mask data into internal register.	4 - 48
SVMLMT	Saves the mask data into memory card.	4 - 48
SVMS	Saves the current status data into memory card.	4 - 45
SVMTRA	Saves the Trace A into memory card.	4 - 46
SVMTRB	Saves the Trace B into memory card.	4 - 47
SVS	Saves the current status data into internal register.	4 - 45
SVTRA	Saves the Trace A in the specified trace register.	4 - 46
SVTRB	Saves the Trace B in the specified trace register.	4 - 47
TDLY	Sets the trigger delay time.	4 - 25
TDF	Trace data format	4 - 37
TGAFC	Tracking Generator Auto Frequency Calibration	4 - 91
TGEN	Tracking Generator Power	4 - 90
TGLEV	Tracking Generator Output Level	4 - 90
TGMFC	Tracking Generator Manual Frequency Calibration	4 - 92
TGNORM	Tracking Generator Normalize	4 - 91
TF	Selects the trigger filter.	4 - 24
TH	Sets the threshold.	4 - 30

TIME	Sets the built - in clock to the specified time.	4 - 80
TITLE	Screen title entry.	4 - 31
TLV	Sets the trigger level.	4 - 24
TM	Sets the trigger switch and trigger source.	4 - 23
TMPCAL	Executes temperature calibration.	4 - 76
TRA/TRB	Trace data input/output	4 - 35
TRAALL/TRBALL	Trace all data output	4 - 36
TRGSWP	Selects the continuous - sweep or the single - sweep mode.	4 - 23
TRSTS	Sets the trace status.	4 - 37
VB	Sets the video bandwidth.	4 - 27
VIEW	Displays the current contents of the selected trace.	4 - 33
WIN	Sets the Window function on or off.	4 - 56
WINLW	Selects the lower window area.	4 - 57
WINLZ	Changes the lower window area to the entire display.	4 - 58
WINT	Toggles the display between the upper window area and lower window area.	4 - 58
WINUP	Selects the upper window area.	4 - 56
WINUZ	Changes the upper window area to the entire display.	4 - 57
XDBCTN	Continuous sweep and X dB measurement.	4 - 61
XDBDW	Places left and right marker at X dB down.	4 - 59
XDBEND	Stop the X dB measurement function	4 - 61
XDBLW	Places the marker at X dB down to the left.	4 - 59
XDBRA	Returns the amplitude result of the X dB Measurements.	4 - 62
XDBRF	Returns the frequency result of the X dB Measurements.	4 - 62
XDBRW	Places the marker at X dB down to the right.	4 - 60
XDBSEL	Selects the X dB marker display format.	4 - 60
XDBSGL	Single sweep and X dB measurement.	4 - 61
YIGCAL	Executes First local oscillator Calibration.	4 - 75
ZI	Zoom - in(Changes to 1/2 the previous span)	4 - 10
ZO	Zoom - out(Changes to two times of the previous span)	4 - 10
ZS	Selects zero frequency span	4 - 9

〈BLANK〉

APPENDIX - ERROR CODE

ERROR CODE		ERROR DESCRIPTION
100	:	Data out of range.
101	:	Center frequency out of range.
102	:	Start frequency out of range.
103	:	Stop frequency out of range.
104	:	Center frequency step size out of range.
105	:	Span frequency out of range.
106	:	Reference level out of range
107	:	Attenuator level out of range.
108	:	Marker function out of range.
111	:	RBW out of range.
112	:	VBW out of range.
113	:	Sweep time out of range.
114	:	Contrast level out of range.
118	:	Threshold level error.
119	:	" LG "data out of range.
120	:	ESE data out of range.
121	:	ESE2 data out of range.
122	:	SRE data out of range.
123	:	Trigger Level data out of range.
124	:	Trigger Level, Trigger Source is not Video.
125	:	Trig Delay data out of range.
126	:	Trig Delay, Span is not Zero.
127	:	Average Count out of range.
128	:	Audio Level out of range.
129	:	Squelch Level out of range.
130	:	Date out of range.
131	:	Time out of range.
132	:	Save internal error.
133	:	Save external error.
134	:	Recall internal error.
135	:	Recall external error.
136	:	PCMCIA error.
137	:	Limit error.
138	:	dB down error.

ERROR CODE	ERROR DESCRIPTION
139	: OBW error.
140	: TRA out of range.
141	: TRB out of range.
142	: Delta marker function error.
143	: Normal marker function error.
144	: Display Line error.
145	: Marker Counter Resolution out of Range.
146	: Noise Marker error.
147	: Printer not connected or not responded.
148	: Printer off - line or paper empty.
149	: PCMCIA option not installed.
150	: Auto SetUp, signal not found.
151	: Peak search error.
152	: RBW CAL Fail.
153	: PWR On CAL Fail.
154	: Temp. CAL Fail.
155	: Tracking Generator is not ON
156	: Tracking Generator output level error
157	: Tracking Generator manual frequency CAL. offset frequency range error
158	: Span Cal Fail
159	: Level Cal Fail
160	: Log Cal Fail
161	: AM scale range over
162	: not AM mode.
163	: FM scale range over
164	: not FM mode
165	: Channel Bandwidth or Space range over in ACP
166	: ACP error
167	: Channel Bandwidth range over in Channel Power
168	: Channel Power error
169	: Reference Level Offset range over
999	: Missing suffix and invalid suffix.
998	: Unnecessary suffix insertion.
997	: Bad command.
996	: Input data sizeover error.
995	: Processing the other function.

IFR AMERICAS
10200 West York Street
Wichita, KS 67215
Phone : 316-522-4981 **Email : @ifrsys.com**
FAX : 316-524-2623 **Web site : www.ifrinternational.com**

IFR LIMITED
Longacres House
Norton Green Road
Stevenage, Herts
England SG1 2BA
Phone : 44 (0) 1438 742200 **Email : @ifrinternational.co.uk**
FAX : 44 (0) 1438 742010 **Web site : www.ifrinternational.com**

IFR Americas, Inc.

IFR AMERICAS
10200 West York Street
Wichita, KS 67215
Phone : 316-522-4981 **Email : @ifrsys.com**
FAX : 316-524-2623 **Web site : www.ifrinternational.com**

IFR LIMITED
Longacres House
Norton Green Road
Stevenage, Herts
England SG1 2BA
Phone : 44 (0) 1438 742200 **Email : @ifrinternational.co.uk**
FAX : 44 (0) 1438 742010 **Web site : www.ifrinternational.com**

IFR Americas, Inc.