MS4630B
Network Analyzer
Operation Manual
Vol. 2 Remote Operation

Fourth Edition

To ensure that the equipment is used safely, read the "For Safety" in the MS4630B Operation Manual Vol.1 first.
Keep this manual with the equipment.

ANRITSU CORPORATION

Document No.: M-W1535AE-4.0

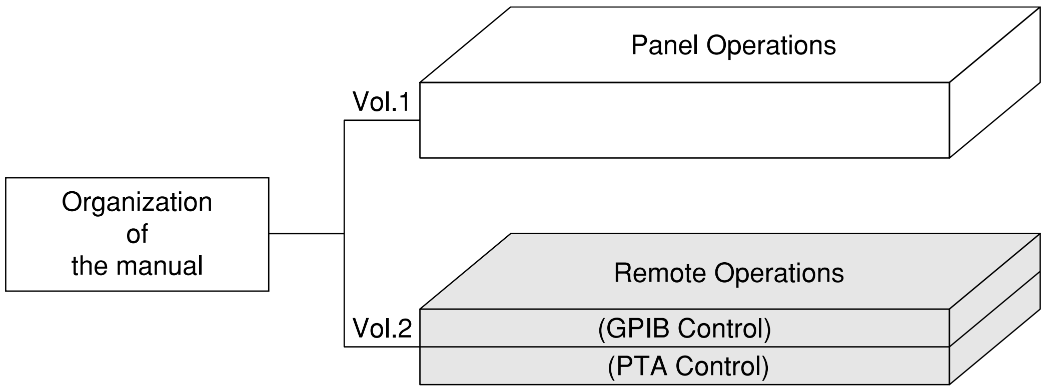
MS4630B
Network Analyzer Vol. 2 Remote Operation
Operation Manual

19 October 1998 (First Edition)
3 February 2003 (Fourth Edition)

About This Manual

Organization of this manual

The manual of this unit is organized in the following two parts (Vol. 1 and Vol. 2).



Panel Operations (Vol. 1):	How to operate the panel of this unit is explained in detail.
Remote Operations (Vol. 2):	This part is organized in GPIB Control and PTA Control. GPIB Control explains about the GPIB commands. PTA Control explains about the PTA operations and commands.

Remote Operations   Operation Manual

1. MS4630B Network Analyzer
   Remote Operations (GPIB Control)
   Operation Manual

2. MS4630B Network Analyzer
   Remote Operations (PTA Control)
   Operation Manual

MS4630B
Network Analyzer
Remote Operations (GPIB Control)
Operation Manual

Contents

Section 6 List of Device Messages

Section 1 Outline

This chapter gives an outline of remote control and describes a system upgrade example.

1.1 Overview

Measurements can be automated by combining the MS4630B Network Analyzer with an external controller (such as a host computer or PC). For this purpose, this unit has the GPIB interface bus (IEEE std 488.2-1987) and RS-232C interface port: option 02.

1.1.1 Remote control function

This unit has the following remote control function and functions related to the remote control.

(1) Controlling the measurement functions except part of the functions such as the power switch and the LOCAL key

(2) Reading the set conditions and measured values

(3) Setting the RS-232C interface conditions from the panel

(4) Setting the GPIB interface conditions from the panel

(5) Selecting the purposes of the interface port from the panel

(6) Configuring an automatic measurement system in combination with PC or other measuring instruments.

1.1.2 Interface port purpose

This unit has, as an interface port with an external device, the GPIB interface as a standard feature, and optionally the RS-232C interface (option 02) and parallel (Centro) interface (option 02). The purpose of these interface ports are showed as follows.

Connection with an external controller: Select from GPIB/RS-232C
Connection port with a printer: Select from GPIB/RS-232C/Centro
Connection port with an external device controlled from PTA: Select from GPIB/RS-232C/Centro

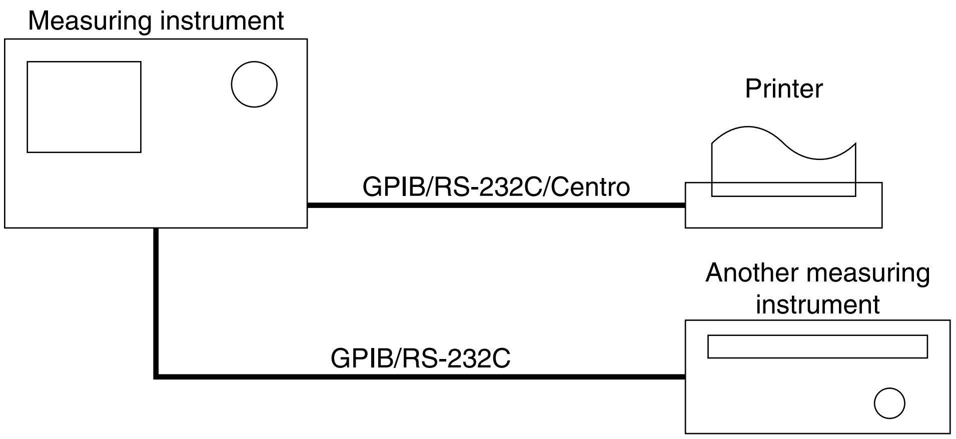
1.1.3 System upgrade example using GPIB/RS-232C

(1) Standalone system (part 1)

Waveforms measured by this unit are output to the printer.

(2) Standalone system (part 2)

Another measuring instrument and the printer are controlled from PTA. The same interface cannot be selected for the connection port with the printer and that for the external device controlled from PTA.

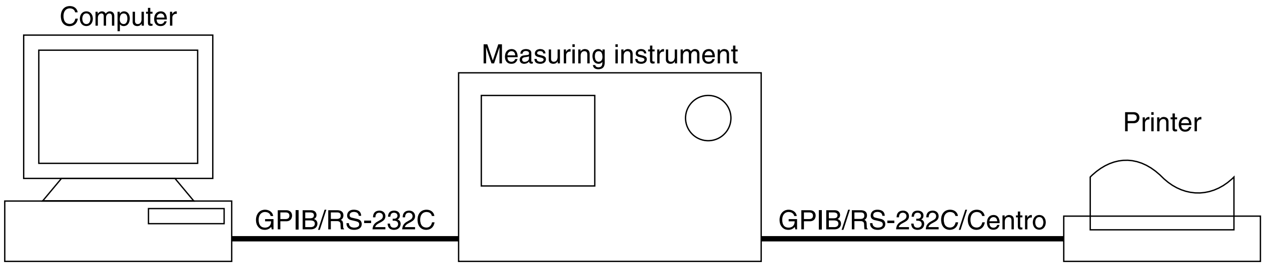


(3) Host computer control (part 1)

The measuring instrument is controlled automatically/remotely from the computer.

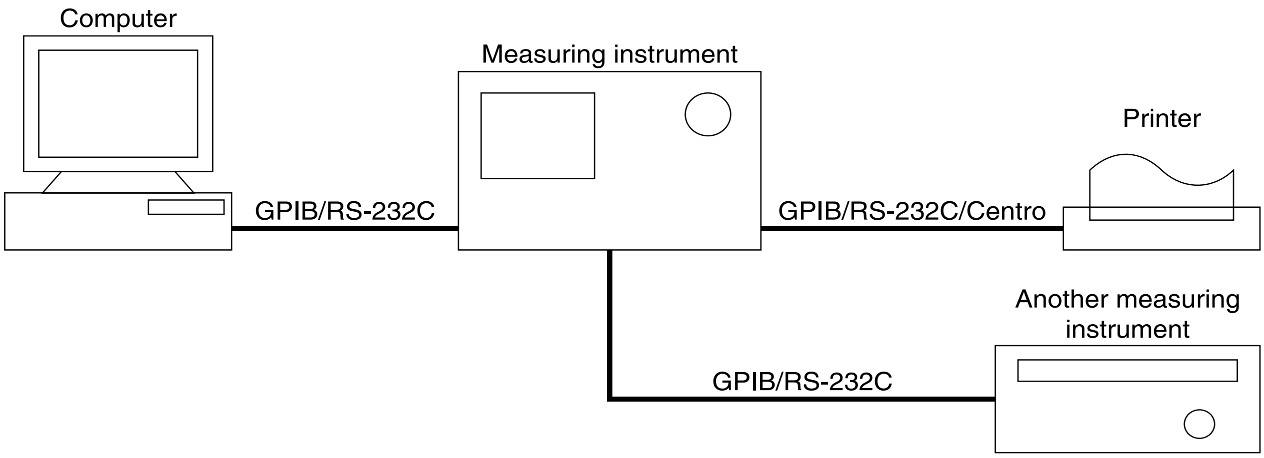
(4) Host computer control (part 2)

The measuring instrument is controlled automatically/remotely from the computer, and measured waveforms are output to the printer. The same interface cannot be selected for the connection port of the external controller and that of the printer.



(5) Host computer control (part 3)

The measuring instrument is controlled automatically/remotely from the computer, and measured waveforms are output to the printer. PTA programs are executed from the computer. The same interface cannot be selected for the connection port of the external controller, that of the printer, and that of the external device controlled from PTA.



1.1.4 GPIB specification

The following shows the GPIB specification of this unit.

Item	Specified value and supplementary explanation
Function	Complying with IEEE488.2 This unit is, as a device, controlled by an external controller (excluding the power switch). This unit controls, as a controller, an external device (such as a printer).
Interface function	SH1 : All source handshake functions present Used for data send timing AH1 : All acceptor handshake functions present Used for data receive timing T6 : Basic talker function and serial polling functions present. No talk-only function. Talker release function by MLA present. L4 : Basic listener function present. No listen-only function. Listener release function by MLA present. SR1 : All functions of Service Request and Status Byte present RL1 : All remote/local functions present Local lockout function present PP0 : No parallel polling function DC1 : All device clear functions present DT1 : Device trigger function present C1 : System controller function present C2 : IFC sending present C3 : REN sending function present C4 : Response to SRQ present C28 : Interface message sending present E2 : Try state output

1.1.5 RS-232C specification

The following shows the specification of RS-232C (option 02) of this unit.

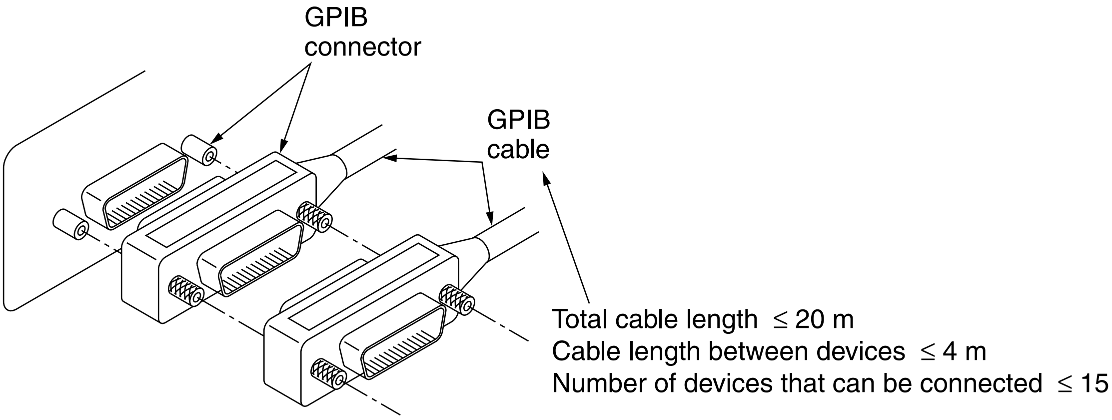
Item	Specified value
Function	This unit is, as a device, controlled by an external controller This unit outputs, as a controller, print data to a printer.
Communication method	Asynchronous (start-stop synchronization), half-duplex
Communication control method	CTS/RTS control
Baud rate	2400, 4800, 9600 bps
Data bit	7 bits, 8 bits
Parity	Odd (ODD), even (EVEN), none (NON)
Start bit	1 bit
Stop bit	1 bits, 2 bits
Connector	D-sub 9 pin, female

Section 2 Connection Method

This chapter explains the RS-232C and GPIB cable connection with external devices such as a host computer, PC, or printer, and how to set the interface of this device.

2.1 Connection Through GPIB Cable

The GPIB connector on the rear side of this device and that of an external device by using the GPIB cable are connected.



Note:

Be sure to connect the GPIB cable before turning on the power of this unit.

Up to 15 devices including the controller can be connected to a system. Follow the conditions described above when connecting the devices.

2.2 Setting the GPIB Interface Conditions

Set the GPIB interface conditions for this unit with the following operations.

(1) Press Local key, or Shift and Control Copy keys.

(2) Press the F1: GPIB key on the menu.
A window for setting the GPIB interface conditions is displayed.

(3) Set each interface condition following the explanations below.

(a) GPIB MY ADDRESS
→The GPIB address of this unit is set (0 to 30).

(b) CONTROL FUNCTION
→The control condition of GPIB of this unit is set.

DEVICE:
This unit operates as a device among the GPIB connected devices.
Set this condition if you want to control this unit from an external computer.
You cannot set the "DEVICE" condition for CONTROL FUNCTION of both RS-232C and GPIB.

CONTROLLER:
This unit operates as a system controller among the GPIB connected devices.
Set this condition if you want to hardcopy to a printer or control external devices from PTA.

(c) ENABLE REGISTER ALL
→ENABLE REGISTER (registers set by *SRE, *ESE, ESE1, ESE2, ESE3) is set.
OFF : All Enable registers are set to OFF (disable state).
ON : All Enable registers are set to ON (enable state).

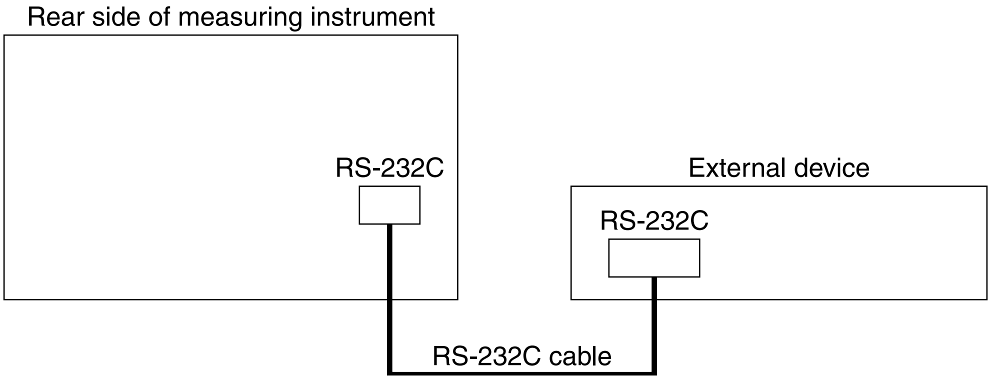
(d) TERMINATOR (for TALKER)
→Terminator code for TALKER data is set.

(e) TIME-OUT
→Time-out time for send-receive operations is set if the "CONTROLLER" condition is selected for CONTROL FUNCTION (0 to 99999 s).
If you set 0, time-out is not set (infinite wait).

(f) ACTIVE PORT or HADR COPY
→Active ports are selected from the interface ports for which the "CONTROLLER" condition is set for CONTROL FUNCTION.
The parallel (Centro) interface operates always in the "CONTROLLER" condition.
An active port is an interface port which actually performs input/output operations for hard copy, or control from PTA.
This function is invalid for interfaces for which the "CONTROLLER" condition is not set.

2.3 Connection with External Devices through RS-232C Cable

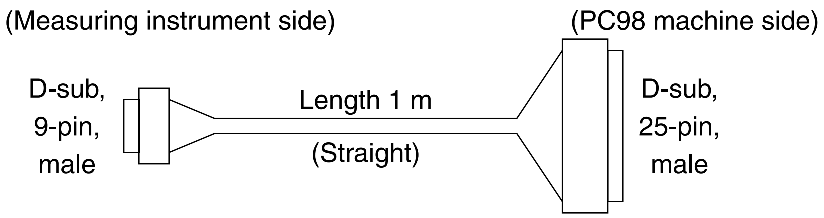
If the RS-232C/Centro interface (option 02) is installed on this unit, the RS-232C connector (D-sub, 9-pin, female) on the rear side of this unit and that of an external device can be connected using an RS-232C cable.



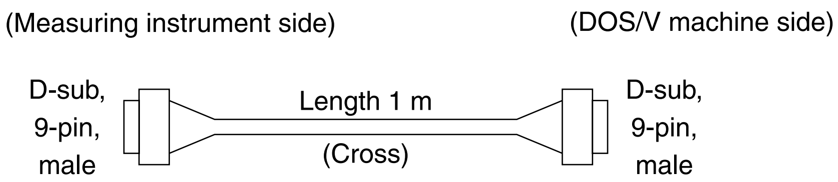
Note:

Since there are two types of RS-232C connectors with a different number of pins (9-pin and 25-pin), it is necessary to check the number of pins of the external device before purchasing an RS-232C cable.

- RS-232C cable (for PC98 machine, VP-600 printer)

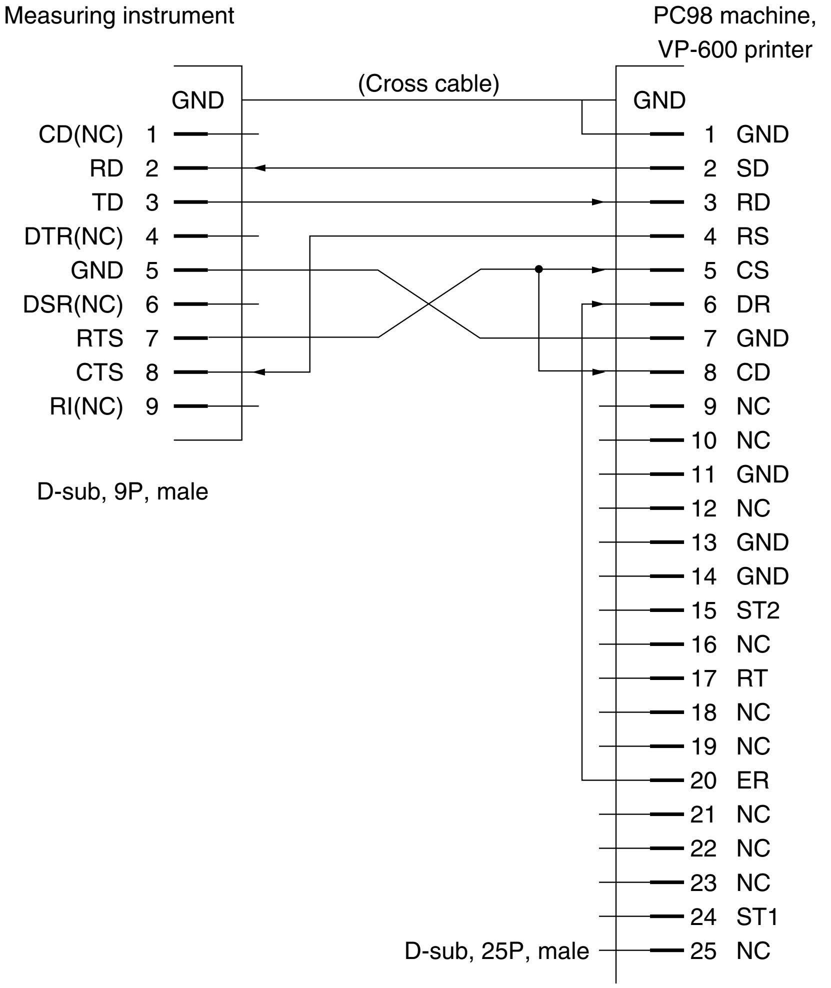


- RS-232C cable (for DOS/V machine)

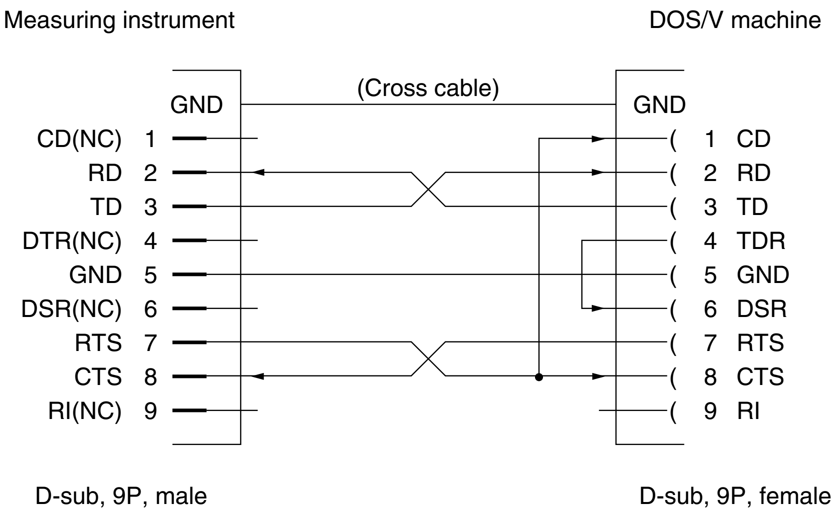


2.4 Connection Diagram of RS-232C Interface Signals

The following figure shows a connection diagram of RS-232C interface signals between this unit, and a PC or a printer.



Connection diagram with a PC98 machine, or VP-600 printer



Connection diagram with a DOS/V machine

2.5 Setting the RS-232C Interface Conditions

Set the RS-232C interface conditions for this unit with the following operations.

(1) Press [Local] key, or [Shift] and [Copy (Control)] keys.

(2) Press the F2: RS232C (Opt) key on the menu.
A window for setting the RS-232C interface conditions is displayed.
The window will not be displayed if option 02 (RS-232C/Centro interface) is not installed.

(3) Set each interface condition following the explanations below.

(a) CONTROL FUNCTION
→The control condition of RS-232C of this unit is set.

DEVICE:
Set this condition if you want to control this unit from an external computer.
You cannot set the "DEVICE" condition for CONTROL FUNCTION of both RS-232C and GPIB.

CONTROLLER:
Set this condition if you want to hardcopy to a printer or control external devices from PTA.

(b) TERMINATOR (for TALKER)
→Terminator code for TALKER data is set.

(c) TIME-OUT
→Time-out time for send-receive operations is set if the "CONTROLLER" condition is selected for CONTROL FUNCTION (0 to 99999 s).
If you set 0, time-out is not set (infinite wait).

(d) ACTIVE PORT for HARD COPY
→Active ports are selected from the interface ports for which the "CONTROLLER" condition is set for CONTROL FUNCTION.
The parallel (Centro) interface operates always in the "CONTROLLER" condition.
An active port is an interface port which actually performs input/output operations for hard copy, or control from PTA.
This function is invalid for interfaces for which the "CONTROLLER" condition is not set.

(e) BAUD RATE
→Set the baud rate (9600/4800/2400 bps).

(f) DATA BITS
→Set the data bit length (7/8 bits).

(g) STOP BIT
→Set the stop bit length (1/2 bits).

(h) PARITY
Set the parity (OFF/EVEN/ODD)

Section 3 Device Message Format

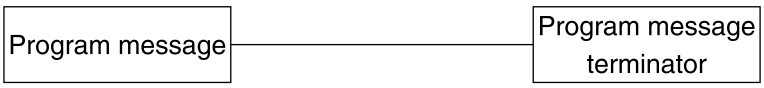
This chapter explains the format of device messages sent and received between the controller (host computer) and device (measuring instrument) through GPIB/RS-232C.

3.1 Overview

Device messages are data that are sent and received between the controller and devices. There are two types of device messages: program messages (data output to this unit from the controller) and response messages (data input into the controller from this unit). Some program messages are program commands that set parameters to this unit or instruct processing. Others are program queries that inquire about the contents of parameters or measurement results.

3.1.1 Program message format

The following format is used to output program messages to this unit from the controller program with a WRITE statement.

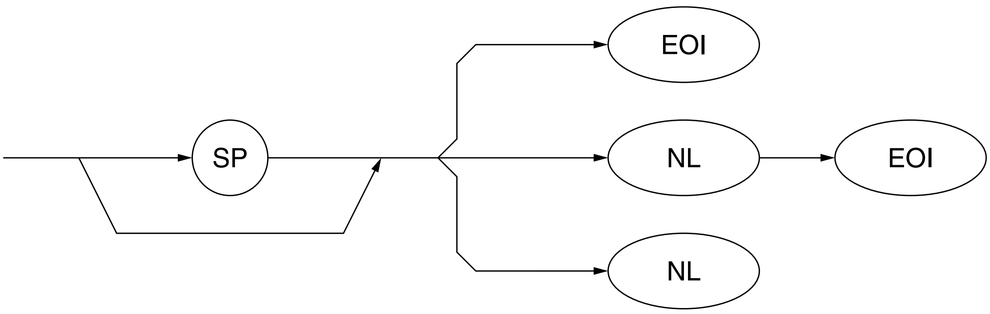



```
WRITE  #1,"CNF 10.7MHZ"
```

Program message

The specified terminator is added when outputting from controller to this unit

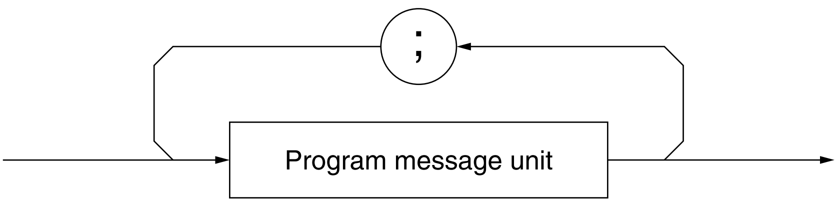
(1) Program message terminator



NL : New Line. Also called LF (Line Feed)

CR (carriage return) is not handled as a terminator and is ignored.

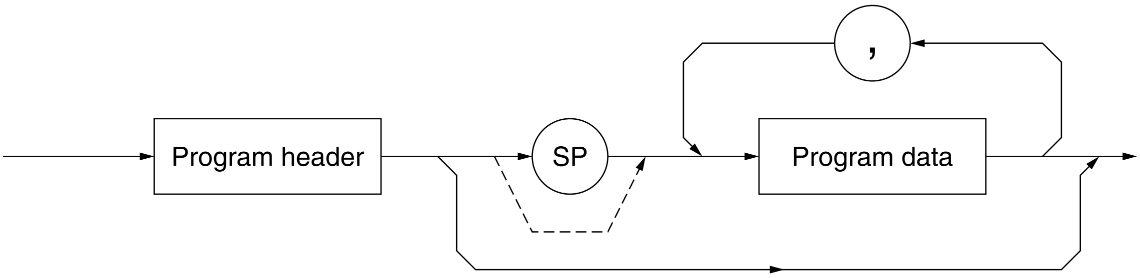
(2) Program message



Multiple commands can be output using ";".

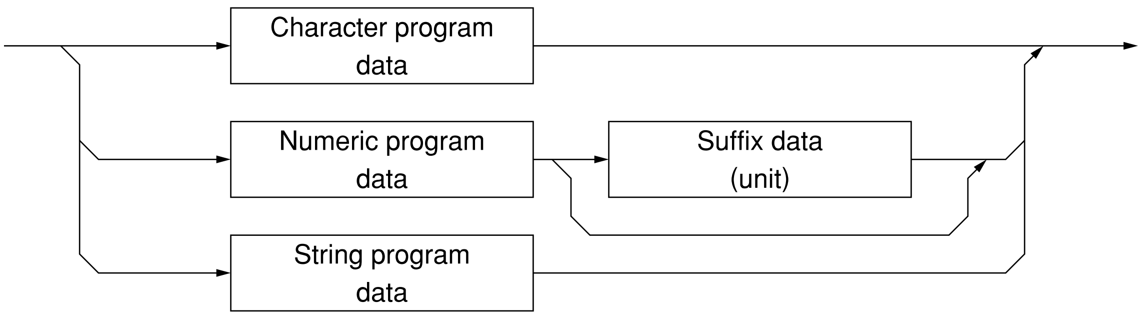
<Example> `WRITE #1, "CNF 10.7MHZ; SPF 100KHZ"`

(3) Program message unit



- "*" is attached to the beginning of the program header of IEEE488.2 common commands.
- If program data is numeric program data, SP between the program header and program data can be omitted.
- The last character of the program header of a program query is generally "?".

(4) Program data



(5) Character program data

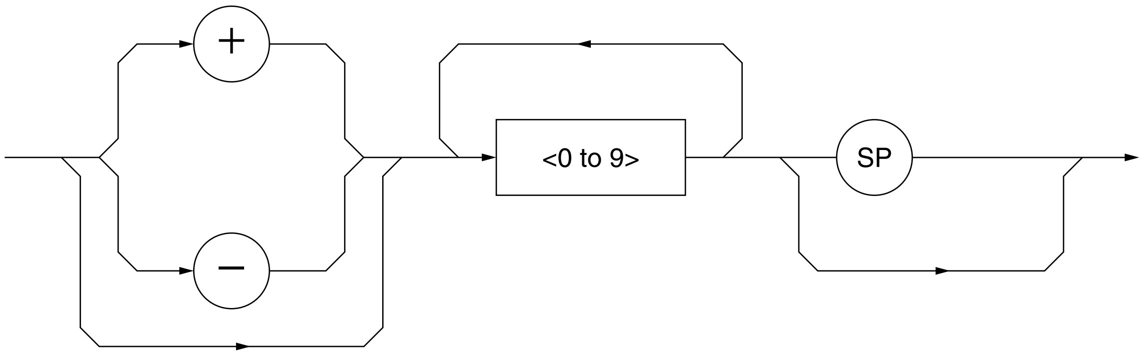
Pre-defined string data consisting of alphabet characters from A to Z.

<Example> `WRITE #1, "CPL MKR, ON"` . . . . The coupled marker function is turned on.

(6) Numeric program data

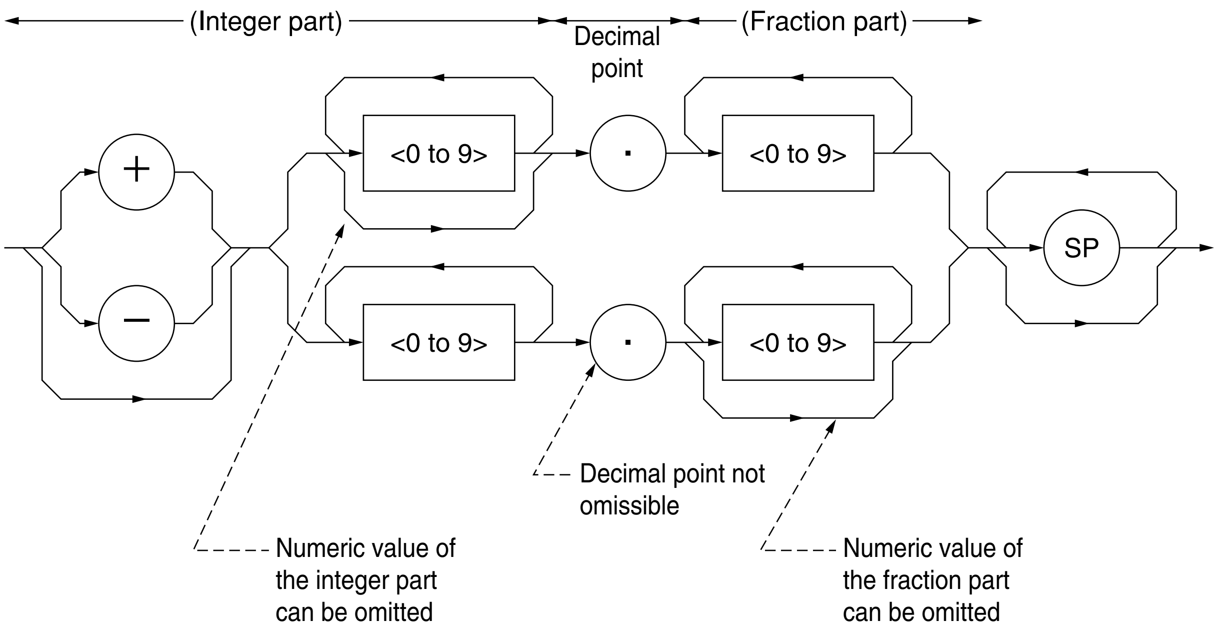
Numeric program data is either in the integer format (NR1) or fixed-point format (NR2).

<Numeric format (NR1)>



- 0 can be set at the beginning. → 005, +000045
- No space is allowed between the sign (+ or –) and numeric character. → +5, +△5 (×)
- Space is allowed after the numeric character. → +5△△△
- The + sign is not obligatory. → +5.5
- The comma cannot be used as a digit separator. → 1,234,567 (×)

<Fixed-point format (NR2)>



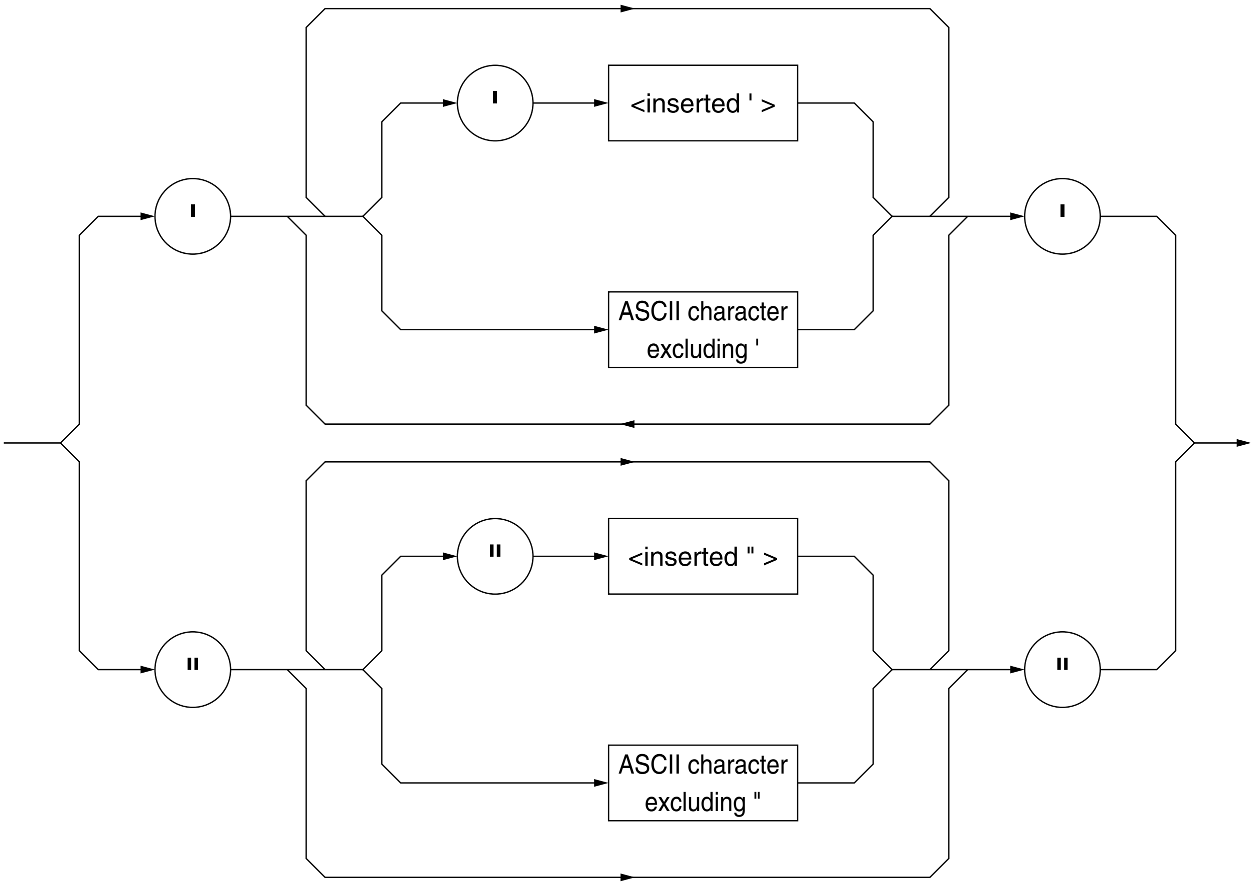
- The numeric representation in the integer format is applied to the (integer part).
- No space is allowed between the numeric character and decimal point. → +753△.123 (×)
- Space can be inserted between numeric characters in the (fraction part). → +753.123△△△
- The numeric value before the decimal point is not obligatory. → .05
- The sign can be put before the decimal point. → +.05, –.05
- The numeric value can terminate with the decimal point. → 12.

(7) Suffix data (units) concerning frequencies

The following table shows the suffix codes (unit codes) used to set measurement parameters and data concerning frequency units.

Frequency unit	Usable suffix code
Setting in hertz units	"HZ" or no suffix code
Setting in kilohertz units	"KHZ", "K" or "KZ"
Setting in megahertz units	"MHZ", "M" or "MZ"

(8) String program data



• A string data is enclosed by a pair of 'such as '.........' .

```
WRITE #1, "TEN 'ABCDE'"
```

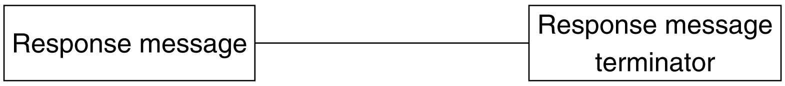
To include ' in a string, specify two ' as ''.

```
WRITE #1, "TEN 'ABCDE''FGH'''"
```

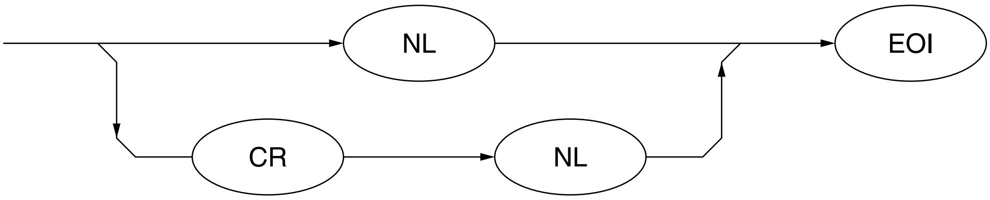
ABCDE'FGH' is set as the title.

3.1.2 Response message format

The following format is used to input response messages to the controller from this unit with a READ statement.

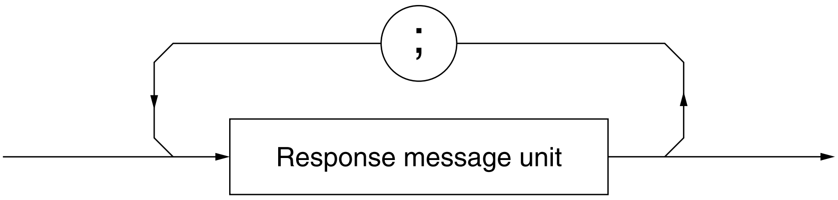


(1) Response message terminator



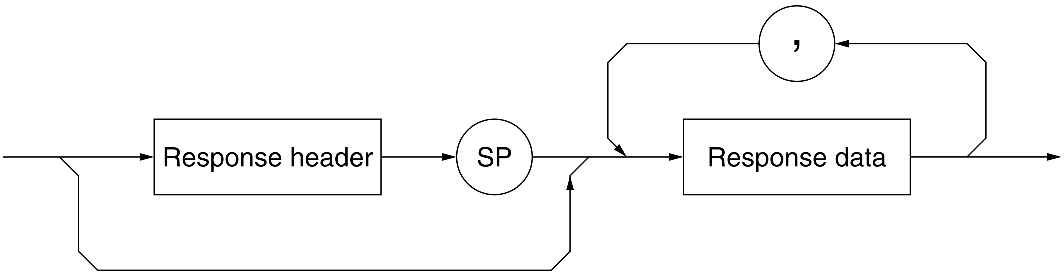
Which of the response message terminators to use can be specified by the TRM command.

(2) Response message

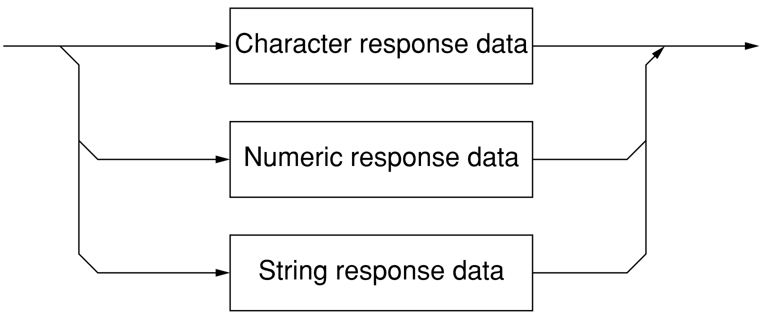


A response message consists of one or more response message units to one or more program queries with one WRITE statement.

(3)Normal response message unit



(4) Response data

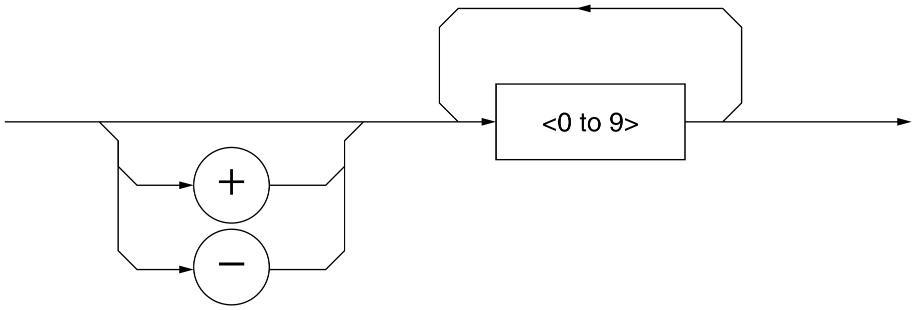


(5) Character response data

Pre-defined string data consisting of A to Z/a to z, 0 to 9, and "_" (underline).

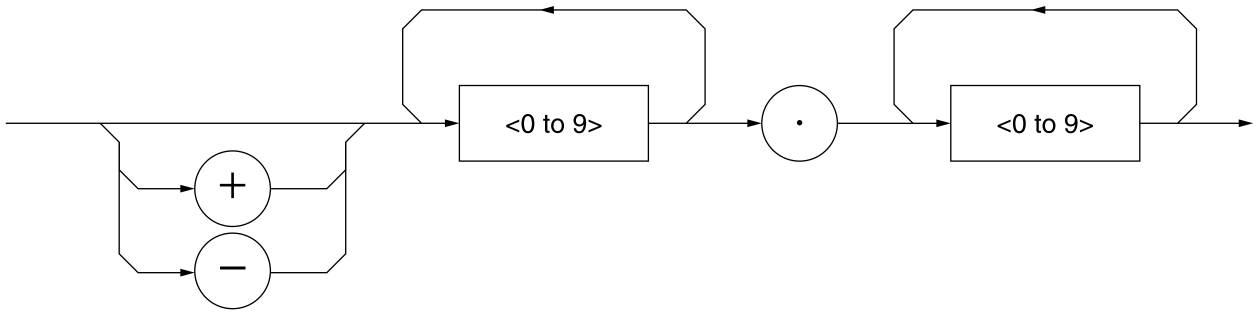
(6) Numeric response data

<Numeric format (NR1)>



- 0 is not allowed as the top-digit numeric character.
  <Example> 123 – 1234

<Fixed-point format (NR2)>



- 0 is not allowed as the top-digit numeric character.
- If the numeric value below the decimal point is 0, the value is output in the integer format.

Section 4 Status structure

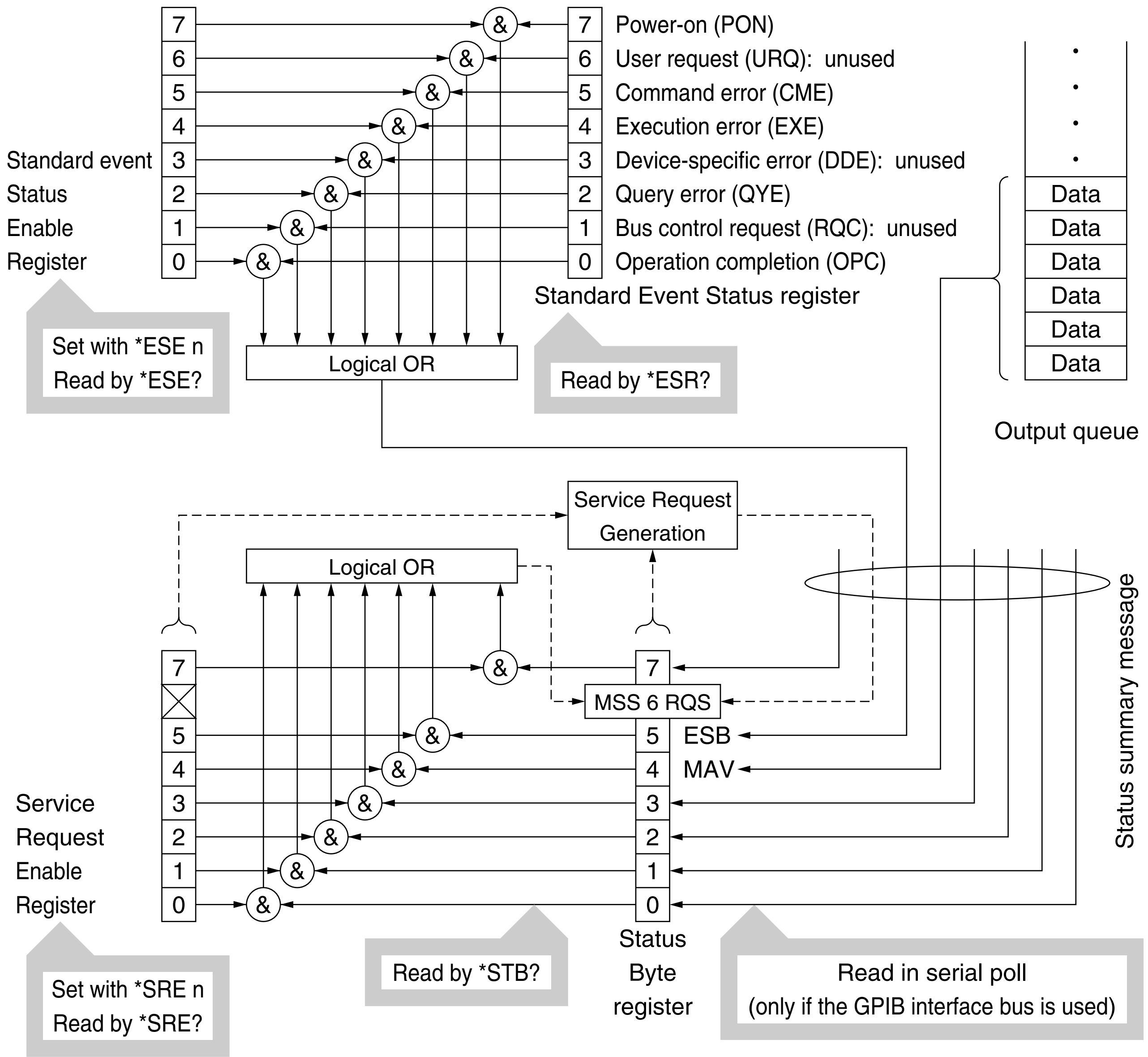
This chapter explains the device status report and its data structure defined in the IEEE488.2 standard for using the GPIB interface. How to synchronize between the device and controller is also explained.

This is a function used for controlling from an external controller using the GPIB interface bus. Except for part of the function, this function can also be used for controlling from an external controller using the RS-232C interface.

The STB-Status Byte sent to the controller is based on the IEEE488.1 standard. Its configuration bits are called the status summary message and summarize the current contents stored in the registers and queues.

4.1 IEEE488.2 Standard Status Model

The following shows a standard model diagram of the status structure defined in IEEE488.2.



Standard status model diagram

In the status model, the IEEE488.1 Status Byte is used as the lowest status. Its Status Byte is composed of seven summary message bits provided by the upper status structure. To generate these summary message bits, the status data structure is composed of the register model and queue model.

Register model	Queue model
The register model is a set of registers to record the events and condition encountered by the device. Its structure is composed of the Event Status Register and Event Status Enable register. If AND of both registers do not produce 0, the corresponding bit of the status bit becomes 1 and otherwise 0. If the result of the logical OR of both registers is 1, the summary message bit becomes 1. If the result of the logical OR is 0, the summary message bit becomes 0.	This is a queue to record the waiting condition or information sequentially. This is called the queue model. In the queue structure, only if there is data in the queue, the corresponding bit becomes 1 and if the queue is free, the corresponding bit becomes 0. Based on the register model and queue model explained above, the standard model of the status data structure of IEEE488.2 is composed of two kinds of register models and one queue model.

Based on the register model and queue model explained above, the standard model of the status data structure of IEEE488.2 is composed of two kinds of register models and one queue model.

(1) Standard Event Status register and Standard Event Status Enable registe
(2) Status Byte register and Service Request Enable register
(3) Output queue

Standard Event Status Register	Status Byte Register	Output Queue
This has the register model structure described above. Among the events encountered by the device, eight kinds of events ((1)power-on, (2)user request, (3)command error, (4)execution error, (5)device-specific error, (6)query error, (7)bus control request, (8)operation completion) are set in the Standard Event Status register as the standard events. The logical OR output bit is represented summarily at Bit 5 (DI06) of the Status Byte register as the Event Status Bit (ESB) summary message.	The Status Byte register is a register in which the RQS bit and seven summary message bits from the status data structure can be set, and is used in combination with the Service Request Enable register. If OR of both registers do not produce 0, SRQ becomes ON. Bit 6 (DI07) of the Status Byte register in this case is reserved as the RQS bit by the system, and this bit notifies the external controller that a service is requested. This SRQ structure is based on the IEEE488.1 standard.	This has the queue model structure described above, and its content notifies that there is data in the output buffer. The content is represented summarily at Bit 4 (DI05) of the Status Byte register as the Message Available (MAV) summary message.

4.2 Status Byte (STB) Register

The STB register is composed of STB of a device and the RQS (or MSS) message.

4.2.1 ESB and MAV summary messages

The following explains about the ESB and MAV summary messages.

(1) ESB summary message

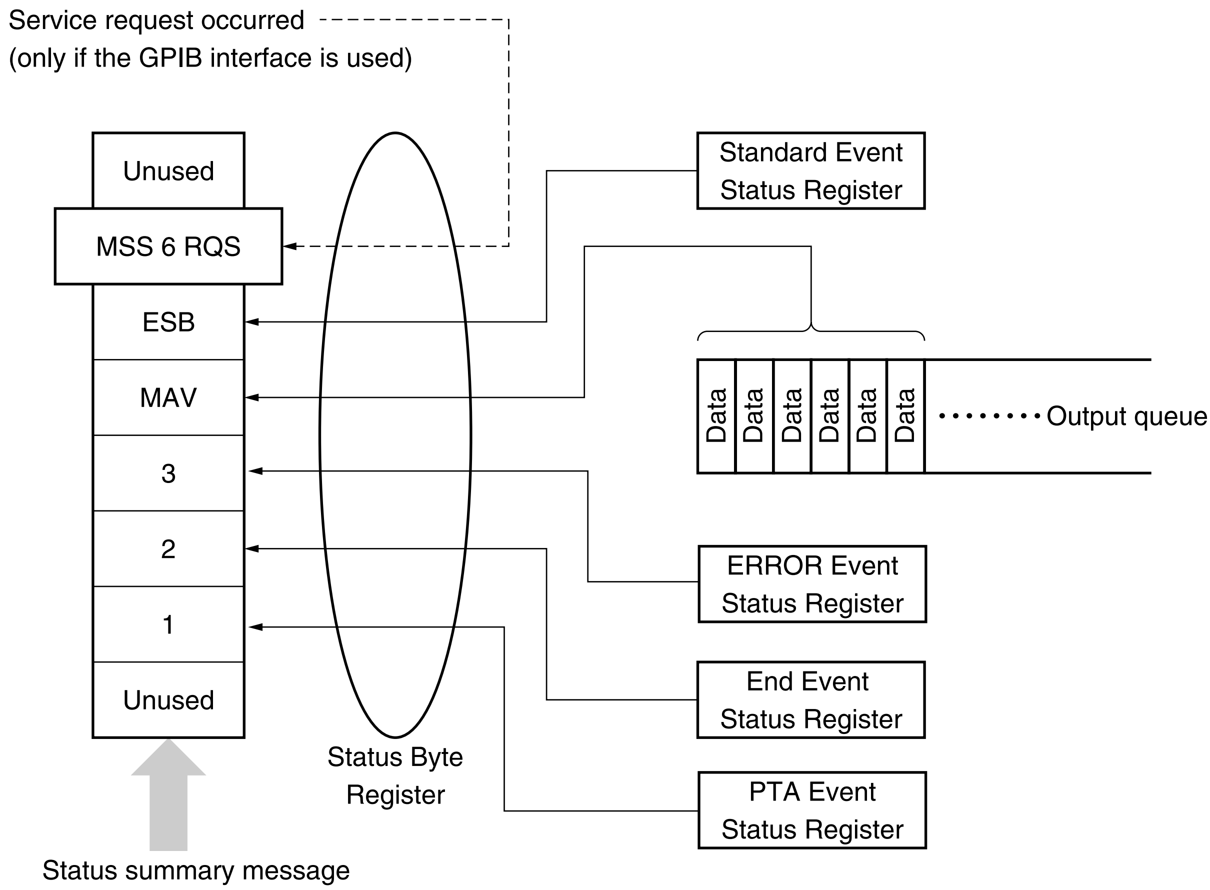
The ESB (Event Summary Bit) message is a message defined in IEEE488.2, and uses Bit 5 of the STB register. The ESB summary message bit becomes 1 if any of the events registered in the Standard Event Status register becomes 1 in a state in which event occurrence is enabled. If no registered event occurs even in a state in which event occurrence is enabled, the ESB summary message bit becomes 0.
If the ESR register is read after an *ESR? query, or the ESR register is cleared by the *CLS command, this bit becomes 0.

(2) MAV summary message

The MAV (Message Available) message is a message defined in IEEE488.2, and uses Bit 4 of the STB register. This bit indicates whether or not the output queue is ‘free’. If a device is ready to accept send requests of response messages from the controller, the MAV summary message bit becomes 1, and if the output queue is ‘free’, it becomes 0. This message is used to synchronize for information exchange with the controller. For example, the controller sends a query command to the device and waits for MAV to become 1. While waiting for a response from the device, the controller can perform other processing. If the output queue is read without checking the MAV, all system bus operations must wait until the device responds.

4.2.2 Device-specific summary message

This device, as shown below, does not use Bit 0 and Bit 7, but uses Bit 1, Bit 2, Bit 3 and Bit 5 as the summary bit for the Event register.



4.2.3 Reading and clearing the STB register

Contents of the STB register are read by using the serial polling or *STB? query. STB messages of IEEE488.1 can be read using either method, but the value sent to Bit 6 (location) depends on the method. Contents of the STB register can be cleared by the *CLS command.

(1) Reading by using the serial polling (only if the GPIB is used)

If serial polling based on IEEE488.1 is carried out, 7 bits of the Status Byte and the RQS message bit based on IEEE488.1 are sent back. The values of the Status Byte are not changed after carrying out serial polling. The device sets the RQS message bit to 0 just after polling is carried out.

(2) Reading by using the *STB common query

Integer-format response messages consisting of the STB register contents and MSS (Master Summary Status) summary messages are sent to the device in response to the *STB common query. Therefore, the response to *STB? is the same as that to serial polling except that the MSS summary message appears at Bit 6 instead of the RQS message.

(3) MSS (Master Summary Status) definition

This message indicates that there is a cause to request at least one service in the device. The MSS message appears at Bit 6 in the response of the device to the *STB query, but does not appear as a response to serial polling. MSS must not be seen as part of the Status Byte of IEEE488.1. MSS is structured by the total OR in the combination of bits of the STB register and SRQ Enable (SRE) register.

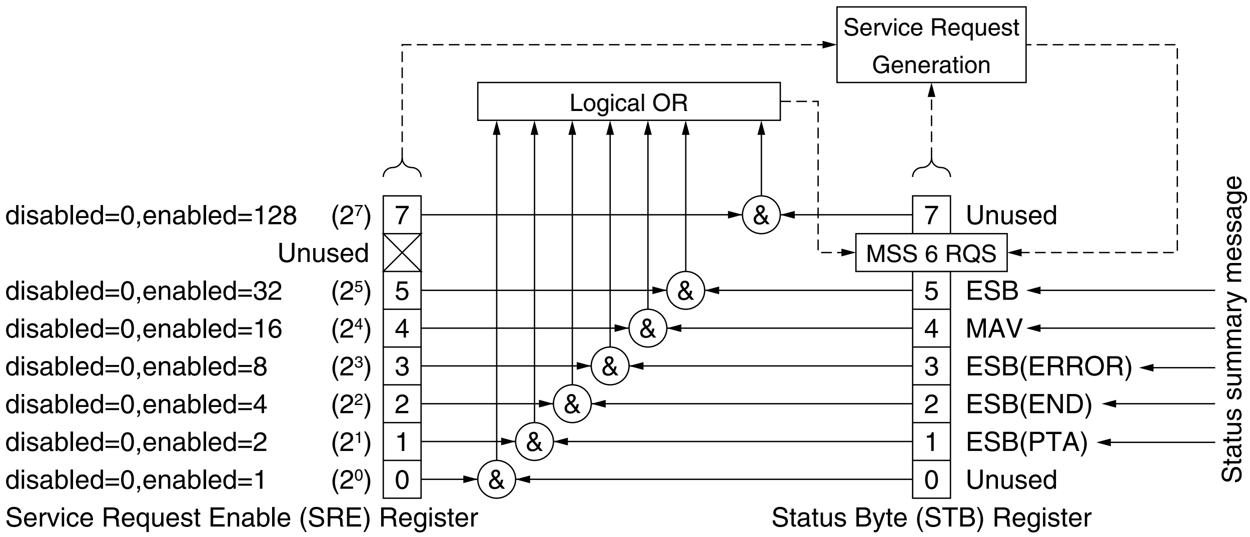
(4) Clearing the STB register by the *CLS common command

The *CLS common command clears all status data structures, and accordingly also clears the corresponding summary messages. The set values of each Enable register are not affected by *CLS.

4.3 Service Request (SRQ) Enable Operation

Whether STB correspondence bits issue SRQ or not can be controlled by the conditions of Bit 0 to Bit 7 of the Service Request Enable (SRE) register.

The bits on the Service Request Enable register correspond to the bits on the Status Byte register. If any of the bits of the Status Byte becomes 1 whose corresponding bits on the Service Request Enable register are set to 1, the device sets the RQS bit to 1, and sends a service request to the controller.



(1) Reading the SRE register

Contents of the SRE register can be read by using the *SRE? common query. The response message to this query is an integer between 0 and 255, and it is the sum of each bit place value of the Service Request Enable register.

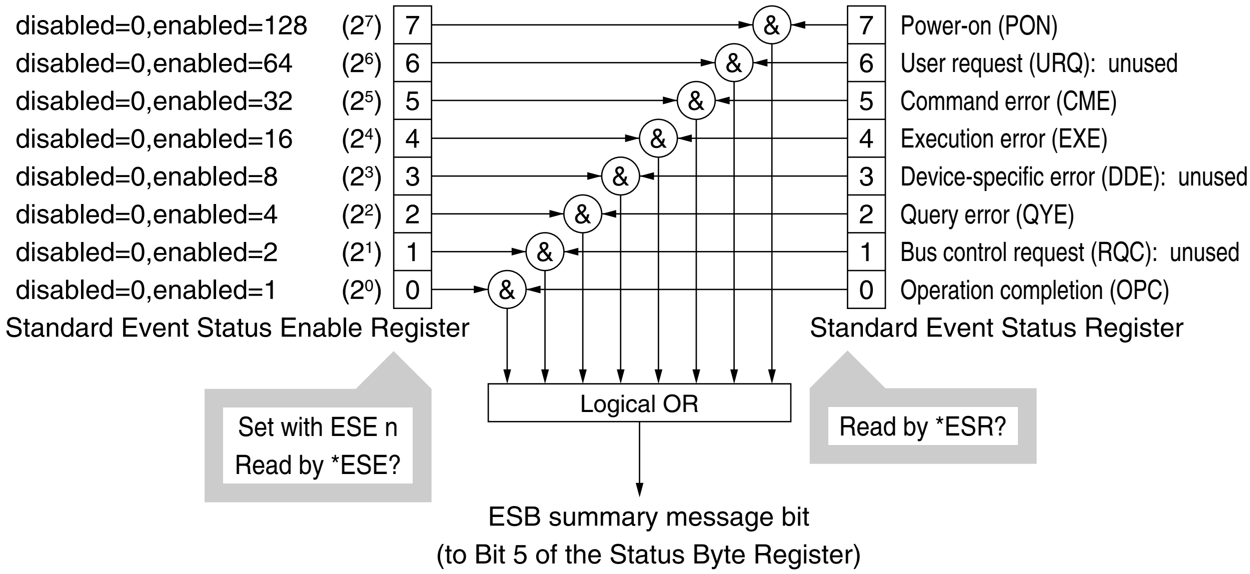
(2) Updating the SRE register

The SRE register can be written in by using the *SRE common command. An integer between 0 and 255 is added as a parameter and the bits of the SRE register are set to 0/1. The value at Bit 6 is ignored.

4.4 Standard Event Status Register

4.4.1 Bit definition for the Standard Event Status register

The following figure shows a Standard Event Status register model.



The Standard Event Status Enable register on the left side selects whether the summary message should be set to True when any bit of the corresponding Event register is set to 1.

Bit	Event name	Explanation
7	Power-on (POP-Power on)	Power changed from OFF to ON.
6	(Unused)	
5	Command error (CME-Command Error)	Grammatically incorrect program message or misspelled command is received.
4	Execution error (EXE-Execution Error)	Non-executable program message, though grammatically correct, is received.
3	(Unused)	
2	Query error (QYE-Query Error)	Attempt is made to read data from the output queue, though there is no data in the output queue. Or, data in the output queue is lost before it is read.
1	(Unused)	
0	Operation completion (OPC-Operation Completion)	This bit becomes 1 when this unit completes the *OPC command.

4.4.2 Reading, writing, and clearing the Standard Event Status Register

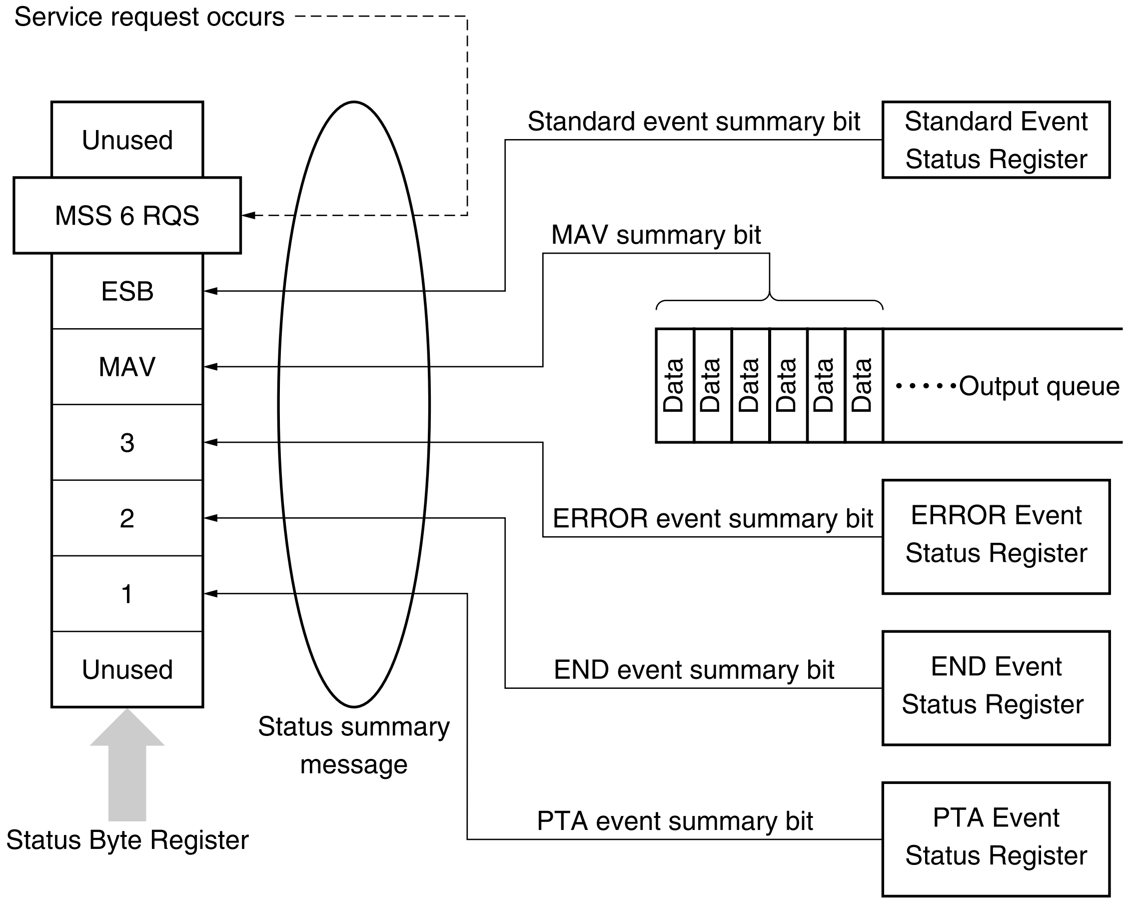
Read	Read by the *ESR? common query. The register is cleared after reading. The response message is integer-format data which is obtained by summing up each event bit with the binary number weight and then converting the sum into a decimal number.
Write	Excluding clearing, nothing can be written from outside.
Clear	In the following cases, the register is cleared. (1) When *CLS command is received. (2) When power is turned on. Bit 7 is set to ON and other bits are set to 0. (3) When an event is read in response to an *ESR query command.

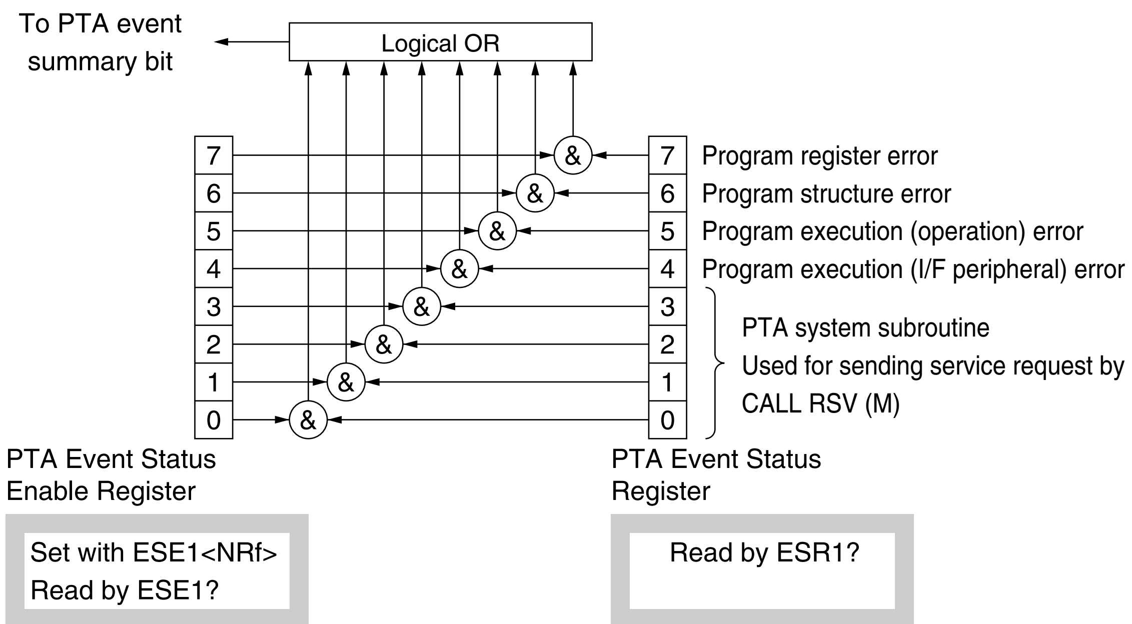
4.4.3 Reading, writing, and clearing the Standard Event Status Enable Register

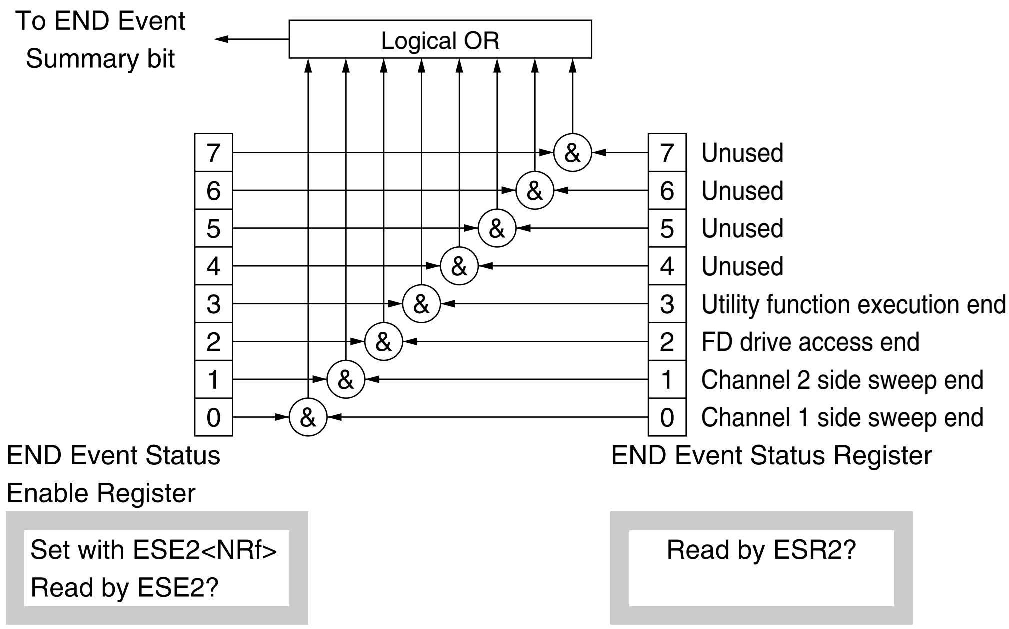
Read	Read by the *ESE? common query. The response message is integer-format data which is obtained by summing up each event bit with the binary number weight and then converting the sum into a decimal number.
Write	*ESE common command is used to write.
Clear	In the following cases, the register is cleared. (1) When *ESE command with a data value 0 is received (2) When power is turned on. The Standard Event Status Enable register is not affected by the following items. (1) Condition change of the device clear function of IEEE488.1 (2) *RST common command reception (3) *CLS common command reception

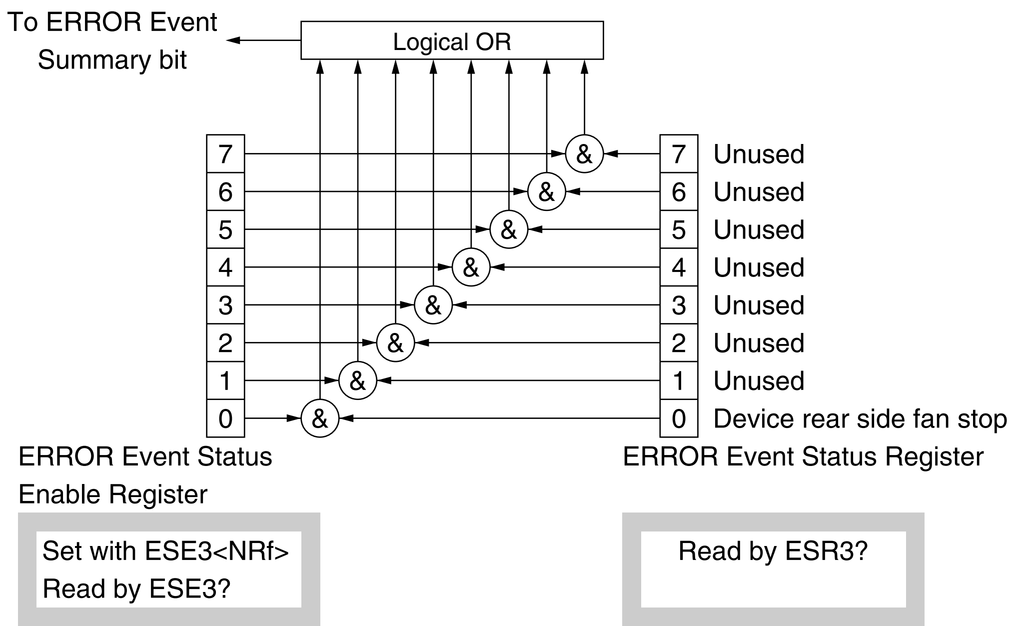
4.5 Extended Event Status Register

In this unit, Bit 7 and Bit 0 are unused and Bit 1 to Bit 3 are allocated for the status summary bits provided by the extended register model.






To END Event Summary bit
Logical OR
7 Unused
6 Unused
5 Unused
4 Unused
3 Utility function execution end
2 FD drive access end
1 Channel 2 side sweep end
0 Channel 1 side sweep end
END Event Status Enable Register
END Event Status Register
Set with ESE2<NRf>
Read by ESE2?
Read by ESR2?


To ERROR Event Summary bit
Logical OR
7 Unused
6 Unused
5 Unused
4 Unused
3 Unused
2 Unused
1 Unused
0 Device rear side fan stop
ERROR Event Status Enable Register
ERROR Event Status Register
Set with ESE3<NRf>
Read by ESE3?
Read by ESR3?

4.5.1 Reading, writing, and clearing the Extended Event Status Register

Read	Read by the *ESR? query. This register is cleared after reading. The response message is integer-format data which is obtained by summing up each event bit with the binary number weight and then converting the sum into a decimal number.
Write	Excluding clearing, nothing can be written from outside.
Clear	In the following cases, the register is cleared. (1) When *CLS command is received. (2) When power is turned on. (3) When an event is read in response to an *ESR? query command.

4.5.2 Reading, writing, and clearing the Extended Event Status Enable Register

Read	Read by the *ESE? query. The response message is integer-format data which is obtained by summing up each event bit with the binary number weight and then converting the sum into a decimal number.
Write	*ESE program command is used to write. Bit 0 to Bit 7 have each the weight 1, 2, 4, 8, 16, 32, 64, and 128. Write data is sent as integer-format data which is obtained by summing up desired bit place values.
Clear	In the following cases, the register is cleared. (1) When *ESE program command with a data value 0 is received (2) When power is turned on. The Extended Event Status Enable register is not affected by the following items. (1) Condition change of the device clear function of IEEE488.1 (2) *RST common command reception (3) *CLS common command reception

4.6 How to Synchronize between Measuring Instrument and Controller

Since this unit handles the specified program messages as sequential commands (after processing of one command is completed, another command is started), there is no need to pay special attention to one-to-one synchronization between this unit and the controller.

If the controller controls multiple devices while synchronizing them, it is necessary to send commands to other devices after all commands specified for this unit are completed.

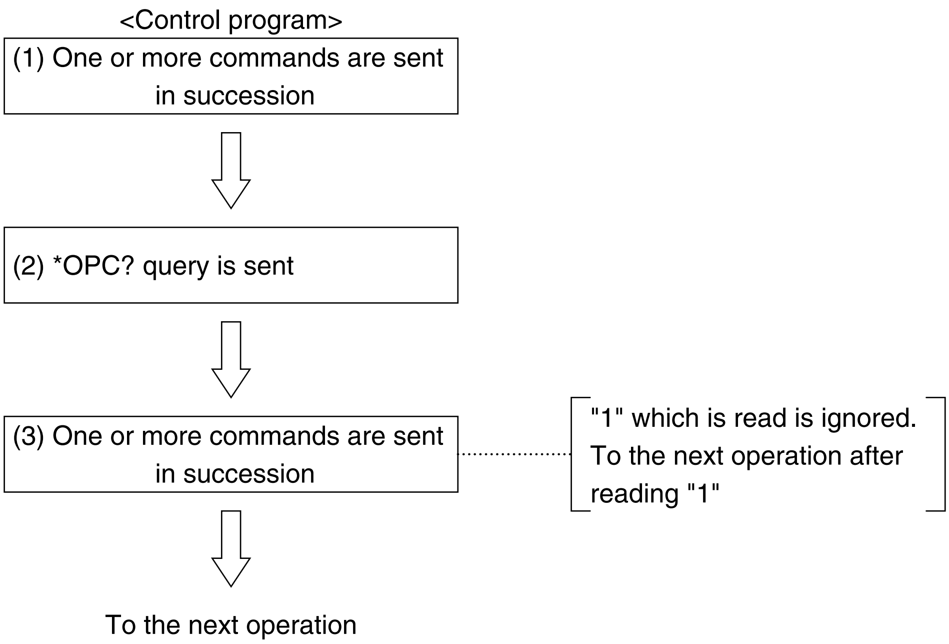
The following two methods are available to synchronize between this unit and the controller.

(1) Response waiting after the *OPC? query

(2) SRQ waiting after *OPC

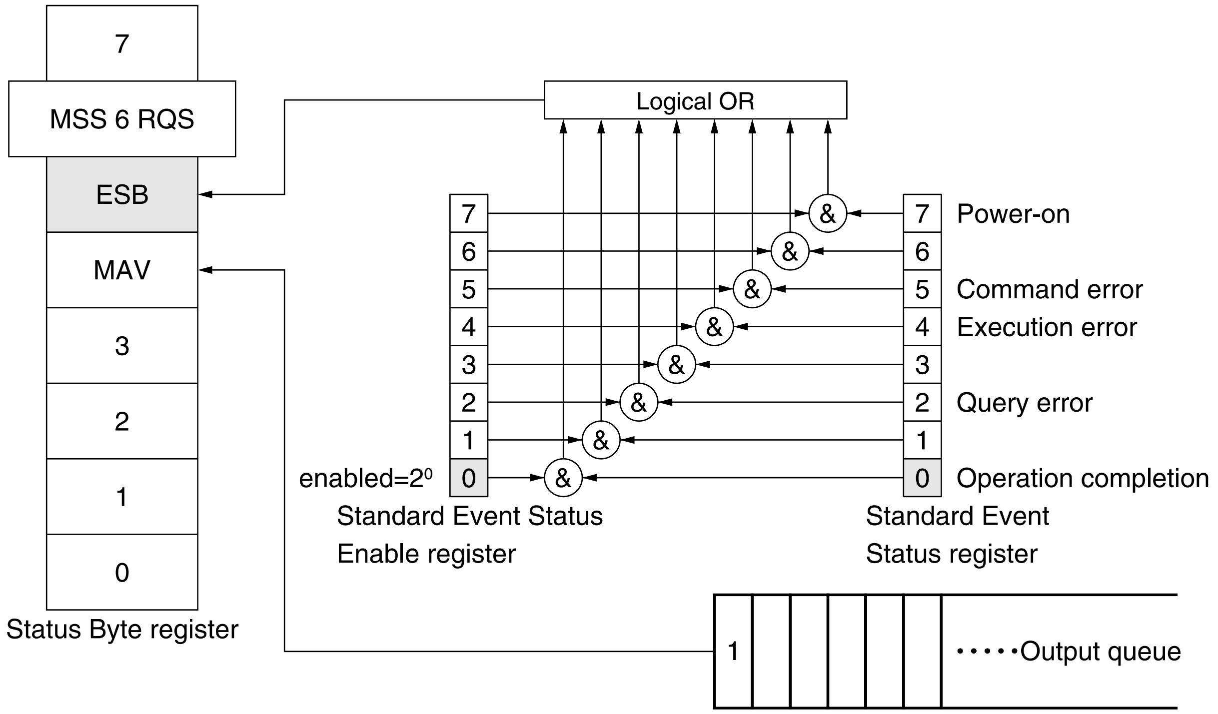
4.6.1 Response waiting after the *OPC? query

If the *OPC? query is performed, this unit outputs “1” as a response message. The controller achieves synchronization by waiting until this response message is input.



4.6.2 Service request waiting after *OPC (only if the GPIB interface bus is used)

If the *OPC command is performed, this unit sets the "Operation completion" bit of the Standard Event Status register to 1. Synchronization is handled by setting this bit to be reflected by SRQ and waiting for it.



■<Control program>

(1) Enable the 2^0 bit of the Standard Event Status Enable register. `PRINT @1;"*ESE 1"`

⇩

(2) Enable the 2^5 bit of the Service Request Enable register. `PRINT @1;"*SRE 32"`

⇩

(3) Execute operations specified for the device.

⇩

(4) Send the *OPC command. `PRINT @1;"*OPC"`

⇩

(5) Wait for SRQ interrupt (ESB summary message) ······Status byte value: $2^6 + 2^5 = 96$

Section 5 Initial Setting

This unit performs initial settings in three levels according to the IEEE488.2 standard. This chapter explains this three-level initialization processing and how to instruct initialization from the controller.

In IEEE488.2, initialization of the GPIB system is divided into three levels. The first level is specified as “bus initialization”, the second level as “message exchange initialization”, and the third level as “device initialization”. Device condition during power-on is also specified so that the condition must be set to a known condition.

Level	Initialization type	Outline	Level combination and order
1	Bus initialization	The interface function of all devices connected to the bus are initialized by the IFC message from the controller.	This level can be used in combination with other levels, but level 1 must be carried out before level 2.
2	Message exchange initialization	The message exchange function for all devices on GPIB is initialized by the GPIB bus command DCL, or that for the specified device is initialized by the GPIB bus command SDC. The function to notify the controller of operation completion is disabled.	This level can be used in combination with other levels, but level 2 must be carried out before level 3.
3	Device initialization	The device specified by the *RST or INI command is restored to its device-specific known condition regardless of the past usage.	This level can be used in combination with other levels, but level 3 must be carried out after level 1 and level 2.

When controlling from the controller using the GPIB interface bus, initialization functions of all levels can be used.

When controlling from the controller using the RS-232C interface bus, level 3, “Device initialization,” can be used. Initialization functions of level 1 and level 2 cannot be used.

In the following, commands for executing the level 1, 2, and 3 initialization and initialization items are explained. Known conditions which are set during power-on are also explained.

5.1 Bus Initialization by IFC Statement

(1) Example

```
board%=0
CALL SendIFC (board%) 2
```

(2) Explanation

This function can be used for controlling from the controller using the GPIB interface bus.
The interface function of all devices connected to the GPIB bus line is initialized by the IFC statement.
Initialization of the interface function is an operation to release the condition (Talker, Listener, and others) of the device interface function set by the controller and restore it to its initial condition, and initializes each function with a circle in the following table. Functions with a triangle are partially initialized.

No	Function	Symbol	Initialization by IFC
1	Source handshake	SH	○
2	Acceptor handshake	AH	○
3	Talker or extended talker	T or TE	○
4	Listener or extended listener	L or LT	○
5	Service request	SR	△
6	Remote/local	RL	
7	Parallel poll	PP	
8	Device clear	DC	
9	Device trigger	DT	
10	Controller	C	○

If the bus is initialized by the IFC statement, device operation conditions (frequency set values, ON/OFF of LED, etc.) are not affected.

5.2 Initialization for Message Exchange by DCL and SDC Bus Commands

(1) Example

Initialization for the message exchange of all devices under the bus (DCL is sent).

```
board%=0
address list%=NOADDR
CALL DevClearList (board%,addresslist%)
```

Initialization for the message exchange of only the third device in the address (SDC is sent).

```
board%=0
address%=3
CALL DevClear (board%,address%)
```

(2) Explanation

This function can be used for controlling from the controller using the GPIB interface bus. This is a statement which initializes the message exchange for all devices on GPIB of the specified select code, or for the specified devices.

(3) Initialization items for the message exchange

Upon receipt of the DCL or SDC bus command, this unit performs the following processing.

1. Input buffer and output queue. .... Both are cleared. The MAV bit is also cleared.
2. Syntax analysis unit, execution control unit, and response creation unit. .... All are reset.
3. Device commands containing *RST. .... Commands which prevent execution of these commands are cleared.
4. *OPC command processing. .... The device is set to the OCIS state (Operation Complete Command Idle state). As a result, the Operation Complete bit cannot be set in the Standard Event Status register.
5. *OPC? query processing. .... The device is set to the OQIS state (Operation Complete Query Idle state). As a result, the operation completion data “1” cannot be set to the output queue.
6. Device function. .... All devices are set to the idle state. The device waits for messages from the controller.

CAUTION ⚠

DCL and SDC bus command processing does not affect the following items.

1. **Current device set data and stored data**
2. **Front panel condition**
3. **Status Byte condition excluding the MAV bit**
4. **Device operations which are currently progressing**

5.3 Device Initialization by the *RST Command

(1) Format

```
*RST
```

(2) Example

In the case of GPIB

```
NWA% = 1
CALL Send (0, NWA, "*RST", NLend)
```

In the case of RS-232C

`WRITE #1, "*RST"` .... Device (this unit) in address 1 is initialized in level 3

(3) Explanation

The *RST (Reset) command is one of the IEEE488.2 common commands and initializes devices in level 3. The *RST (Reset) command is used to set the device (this unit) to a specific initial state. For details of the initialization items and initial values, see Appendix A.

Notes:

The *RST command does not affect the following items.

1. IEEE488.1 interface condition
2. Device address
3. Output queue
4. Service Request Enable register
5. Standard Event Status Enable register
6. Power-on-status-clear flag setting
7. Calibration data affecting the device standard
8. Set parameters related to external device control

5.4 Device Initialization by the INI Command

(1) Format

```
INI
```

(2) Example (program message)

In the case of GPIB

`CALL Send (0, NWA, "INI", NLend)`

In the case of RS232-C

`WRITE #1, "INI"` .... Device (this unit) in address 1 is initialized in level 3

(3) Explanation

The INI command is one of the device messages specific to this unit and initializes devices in level 3. For details of the initialization items and initial values, see Appendix A.

5.5 Device Condition during Power-on

If power is turned on

(1) The condition when power is turned off last time is set.

(2) The input buffer and output queue are cleared.

(3) The syntax analysis unit, execution control unit, and response creation unit are initialized.

(4) The device is set to the OCIS state (Operation Complete Command Idle state).

(5) The device is set to the OQIS state (Operation Complete Query Idle state).

(6) The Standard Event Status register and Standard Event Status Enable register are cleared. Events are registered after clearing.

As a special case of (1), when power is turned on for the first time after shipment, the state of the initial setting list (Appendix A) is reproduced.

Section 6 List of Device Messages

This section lists device messages that can be used for this device according to functions in the order shown in the following table.

MS4630B List of device messages

Function summary	Control command	Query	Response
■ Initialization -PRESET			
Device initialization	INI	*********************	*********************
■ Measurement channel -MENU			
Measurement channel selection	SELCH n 0: CH1&CH2 1: CH1 2: CH2	SELCH?	SELCH n
Active channel selection	ACCH n 1: CH1 / 2:CH2	ACCH?	ACCH n
Coupled channel ON/OFF	CPL CH, ON CPL CH, OFF	CPL CH?	CPL CH, ON CPL CH, OFF
■ Measuring (analytical) port -MEAS			
Analytical port	MEASPT n 0: TB/TA 1: TA/R 2: TB/R 3: TA 4: TB 5: R	MEASPT?	MEASPT n
■ Impedance measurement			
Measuring method selection (SYSTEM)	IMPMD n 1: Reflection (bridge) 2: Transmission (PI-NET)	IMPMD?	IMPMD n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Measuring format -FORMAT			
Measuring format	TRC n 0: LOGMAG 1: PHASE 2: DELAY 3: MAG & PHASE 4: MAG & DELAY 5: POLAR 6: IMPD CHART 7: ADMT CHART 8: VSWR 9: LINMAG 10: LIN & PHASE 11: LIN & DELAY 12: REAL 13: IMAG 14: REAL & IMAG 15: LOG Z 16: LOG Z & θ 17: Q 18: LOG Z & Q	TRC?	TRC n
Impedance marker display form	IMV n 0: Z/θ 1: Rs/Ls, Cs 2: Q/D 3: R+jX	IMV?	IMV n
Admittance marker display form	ADV n 0: Y/θ 1: Rp/Lp,Cp 2: Q/D 3: G+jB	ADV?	ADV n
Phase offset	PHOF r or PHO r r=-180 to +180 (deg)	PHOF? PHO?	PHOF r PHO r

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Frequency -FREQUENCY, SPAN			
Frequency band setting mode	FRQ n 0: CF/SPAN 1: START/STOP	FRQ?	FRQ n
Linear/log sweep switching	LOG n 0: LIN / 1:LOG	LOG?	LOG n
Center frequency	CNF f f=0 to 300 000 000(Hz)	CNF?	CNF f
Frequency span	SPF f f=0 to 300 000 000(Hz)	SPF?	SPF f
Start frequency	STF f In the case of linear sweeping f=0 to 300 000 000(Hz) In the case of log sweeping f=10Hz to 100MHz (1 and 10 steps)	STF?	STF f
Stop frequency	SOF f In the case of linear sweeping f=0 to 300 000 000(Hz) In the case of log sweeping f=100Hz to 100MHz (1 and 10 steps) And 300 MHz	SOF?	SOF f

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Waveform averaging/ smoothing -AVG			
Averaging	AVG r r= 1 to 1000	AVG?	AVG r
Smoothing	SMT r r= 0 to 50 (%) Note	SMT?	SMT r
Delay aperture	HDRG r r= 0.2 to 20 (%)	HDRG?	HDRG r
Resolution bandwidth auto-setting	AU2 n 0: OFF (MANUAL) 1: ON (AUTO)	AU2?	AU2 n
Resolution bandwidth (RBW)	RBW n 0: 3Hz 1: 10Hz 2: 30Hz 3: 100Hz 4: 300Hz 5: 1kHz 6: 3kHz 7: 10kHz 8: 500Hz 9: 2kHz 10: 4kHz 11: 5kHz 12: 20kHz 13: AUTO	RBW?	RBW n
Averaging type	AVT n n 0: SUM 1: MAX HOLD 2: MIN HOLD	AVT?	AVT n

Note:

The setting range depends on the number of measurement points.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Sweep control -SWEEP			
Sweep mode setting	SW2 n 0: REPEAT 1: SINGLE	SW2?	SW2 n
Sweep start/execution			
Starting repeat sweep	SWP 0	SWP?	0:Sweep end
Starting single sweep	SWP 1		(1:Sweeping)
Executing single sweep	SWP 2 Note		No response data is generally returned during sweeping.
Sweep stop/reset/restart	SW3 n 0: STOP 1: RESET 2: CONTINUE		
Sweep time auto-setting	AU1 n 0: OFF(MANUAL) 1: ON(AUTO)	AU1?	AU1 n
Sweep time	SWT t t: 1 to 99 000 000 (ms)	SWT?	SWT t
Total sweep/marker sweep	SW1 n 0: FULL SWEEP 1: MARKER SWEEP	SW1?	SW1 n

Note:

The single sweep start command “SWP 1” only starts the sweep.
The single sweep execution command “SWP 2” both starts and ends the sweep.
Therefore, if “SWP 2” is used to execute the sweep, no further device message can be received until the sweep is ended.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Sweep control -SWEEP			
Measuring point	MEP n 0: 11 1: 21 2: 51 3: 101 4: 251 5: 501 6: 1001	MEP?	MEP n
Breakpoint	BKP r r= 1 to 1001	BKP?	BKP n
External trigger sweep	TRGMD n 0: OFF (INT) 1: ON (EXT)	TRGMD?	TRGMD n
External trigger mode	TRGSW n 0: NORMAL 1: STEP 2: STATE	TRGSW?	TRGSW n
External trigger edge	TRGEG n 0: RISE 1: FALL	TRGEG?	TRGEG n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Output/input -OUT/INPUT			
Output power	OPL r r=0 to 21 (dBm) Note 1	OPL?	OPL r
Source power	SPWR r r=0 to 21 (dBm) Note 1	SPWR?	SPWR r
Output offset	OOFS r r=-100 to +100 (dB)	OOFS?	OOFS r
Output attenuator (Opt 10)	OATT r Note 2 r=0 to 70 (dB) 10dB STEP	OATT?	OATT r
Power sweep ON/OFF	LSW n 0: OFF/1:ON	LSW?	LSW n
Power sweep start power	STL r r=0 to +21 (dBm) Note 1	STL?	STL r
Power sweep end power	SOL r r=0 to +21 (dBm) Note 1	SOL?	SOL r

Note 1:

Indicates the setting range of output port A without an Opt 10 output attenuator.
The setting range varies depending on whether an Opt 10 output attenuator is installed and how the output port is set.

Note 2:

This setting is enabled only if an Opt 10 output attenuator is installed.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Output/input -OUT/INPUT			
Power sweep step level	SEL r r= 0 to +21 (dB)	SEL?	SEL r
Output port switching	OUTCND n 0: port A/1: port B	OUTCND?	OUTCND n
Input range (R)	IAR n 0: 0 dBm/1:+20 dBm	IAR?	IAR n
Input range (TA)	IATA n or IRG n 0: 0 dBm/1:+20 dBm	IATA? IRG?	IATA n IRG n
Input range (TB)	IATB n 0: 0 dBm/1:+20 dBm	IATB?	IATB n
Input impedance (R)	IMPR n 0: 50/75 Ohm 1: 1M Ohm	IMPR?	IMPR n
Input impedance (TA)	IMPTA n 0: 50/75 Ohm 1: 1M Ohm or IMP n n 0: 50/75 Ohm (1: 50/75 Ohm) 2: 1M Ohm	IMPTA? IMP?	IMPTA n IMP n
Input impedance (TB)	IMPTB n 0: 50/75 Ohm 1: 1M Ohm	IMPTB?	IMPTB n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Output/input -OUT/INPUT			
Standard impedance	RIMP r r=0.1 to 10 000 (Ohm)	RIMP?	RIMP r
Input overload state	*********************	OVL?	OVL n n 0: NORMAL 1: OVER LOAD

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Calibration -CALIBRATION			
X → S normalize sweep start	CAL (Sweep start only) CAL 2 (with sweep end synchronization)	CAL?	0:Sweep end (1:Sweeping) No response data is generally returned during sweeping.
X-S normalize function ON/OFF	CXS n 0: OFF/1:ON	CXS?	CXS n
Calibration type selection	ECL n n 0: RESPONSE 1: 1PORT OSL 2: (Undefined) 3: 1PATH 2PORT 4: RESPONSE & ISOLATION 5: PI-NET	ECL?	ECL n
Calibration data uptake sweep start	COR n	COR?	0:Sweep end (1:Sweeping) No response data is generally returned during sweeping.
Response calibration	0: Response data		
1PORT OSL calibration	0: OPEN data 1: SHORT data 2: LOAD data		
1PATH 2PORT calibration	0: THROUGH data 1: OPEN data 2: SHORT data 3: LOAD data		
Response and isolation calibration	0: RESPONSE data 1: ISOLATION data		
PI-NET calibration	0: OPEN data 1: SHORT data		

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Calibration -CALIBRATION			
Calibration ON/OFF	CORR n 0: OFF/1:ON	CORR?	CORR n
Device parameter for calibration			
Open device C0 (E-15)	CC0 r	CC0?	CC0 r
Open device C1 (E-27)	CC1 r	CC1?	CC1 r
Open device C2 (E-36)	CC2 r	CC2?	CC2 r
Open device C3 (E-45)	CC3 r r=-999.999 to +999.999	CC3?	CC3 r
Device parameter for calibration			
Open device offset length	COO r	COO?	COO r
Short device offset length	COS r	COS?	COS r
Through line offset length	COT r r=-9999.99 to +9999.99	COT?	COT r

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Waveform display scale -SCALE			
Auto-scale	SAU	*********************	*********************
Display scale			
Active TRACE side	SCAL n	SCAL?	SCAL n
TRACE-A side	SCA n	SCA?	SCA n
TRACE-B side	SCB n	SCB?	SCB n
dB unit	n=0 to 11 0:0.01dB/div After this, changes are in steps of 1, 2, or 5		
deg unit	n=0 to 11 0:0.01deg/div After this, changes are in steps of 1, 2, or 5		
sec unit, absolute number	n=0 to 11 0:measuring resolution* 100/div After this, changes are in steps of 1, 2, or 5		
Ohm unit	n=1,2,4,5,8,10 DECADE number in logarithmic scale is set		

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Waveform display scale -SCALE			
Display offset			
Active TRACE side	OFST r	OFST?	OFST r
TRACE-A side	OFA r	OFA?	OFA r
TRACE-B side	OFB r	OFB?	OFB r
dB, deg unit	r= -800.000 to +800.000		
sec unit, absolute number	r= measuring resolution × (±8000000) range		
Ohm unit	r= 1m to 100M (Ohm) 1, 10, 100 STEP Scale bottom value in logarithmic scale is set		
Display offset (reference point)	OFS n n 0: center line 1: upper bottom line 2: lower bottom line	OFS?	OFS n
	OFLN r r= 0 to 10 (line)	OFLN?	OFLN r
Electric length compensation	ELG r r=-999 999 to +999 999 (m)	ELG?	ELG r

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Trace (waveform) -TRACE			
Active trace	ACTR n 0: TRACE-A 1: TRACE-B	ACTR?	ACTR n
Split display ON/OFF	DF2 n 0: OFF/1:ON	DF2?	DF2 n
Waveform storage ON/OFF	STOR n 0: OFF/1:ON	STOR?	STOR n
Overwrite display ON/OFF			
Active TRACE side	OVP n	OVP?	OVP n
TRACE-A side	OVPA n	OVPA?	OVPA n
TRACE-B side	OVPB n 0: OFF/1:ON	OVPB?	OVPB n
Grid type	DF1 n 0: ALL 1: CENTER only 2: FRAME only	DF1?	DF1 n
Display item selection	DF4 n	DF4?	DF4 n
Delete item selection	DF3 n n= 1 to 13 Number of concerned item Note	DF3?	DF3 n
DISPLAY ALL function ON/OFF	DAS n 0: OFF (only selected items are displayed) 1: ON (all items are displayed)	DAS?	DSA n

Note:

For the display/delete items, see the subsection “CER and CRN subroutines” in Section 5.2 “System Subroutine” in the Remote Control (PTA Control) manual.
Only if the DISPLAY ITEM function is turned OFF, the selected items can be displayed or deleted.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Trace (waveform) -TRACE			
Subtrace ON/OFF	STR n 0: OFF 1: MT→T 2: MT=ST 3: MT=MT-ST 4: ON	STR?	STR n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Marker -MKR			
Active marker value read			
Active TRACE side	********************	MKV?	value
TRACE-A side	********************	MVA?	value
TRACE-B side	********************	MVB?	value
Multi-marker value read			
Active TRACE side	********************	MKV? 0, n	valueA,valueB
TRACE-A side	********************	MKV? 1, n	valueA
TRACE-B side	********************	MKV? 2, n n=0 to 9: marker number	valueB
Active marker frequency read and setting (frequency)			
Active TRACE side	MKF f	MKF?	f
Multi-marker frequency read and setting (frequency)			
Active TRACE side	MKF 0, n, f	MKF? 0, n	fA, fB
TRACE-A side	MKF 1, n, f	MKF? 1, n	fA
TRACE-B side	MKF 2, n, f n=0 to 9: marker number f: frequency	MKF? 2, n n=0 to 9: marker number	fB
Multi-marker setting and read (point)	MSET 0, n, p n=0 to 9: marker number p=0 to 1000: point	MSET? 0, n	MSET 0, n, p
Multi-marker setting and read (frequency)	MSET 1, n, f n=0 to 9: marker number f: frequency	MSET? 1, n	MSET? 1, n, f
Multi-marker release	MRST n n=0 to 9: marker number	********************	********************

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Marker -MKR			
Reference marker value read			
TRACE-A side	********************	RFA?	valueA
TRACE-B side	********************	RFB?	valueB
Zero marker value read			
TRACE-A side	********************	ZRA?	valueA
TRACE-B side	********************	ZRB?	valueB
Marker ON/OFF state read	********************	MRST? 1	b (10 digits) first digit: marker No. 9 last digit: marker No. 0 0:OFF/1:ON
Active marker No. read	********************	MRST? 2	n: marker No.
Reference marker position setting/read (point)	RMK p p=0 to 1000	RMK?	RMK p
Reference marker number setting	RMKR n n=0 to 9: marker number	RMKR?	RMKR n 00 to 09
Active marker position setting/read (point)	CMK p p=0 to 1000	CMK?	CMK p
Coupled marker ON/OFF	CPL MKR,OFF CPL MKR,ON	CPL? MKR	CPL MKR, OFF CPL MKR, ON
Marker delete	MKD n 0: display/1: delete	MKD?	MKD n
Marker setting mode (SYSTEM)	MKRMD n 0: display position (POINT) 1: frequency (FREQ)	MKRMD?	MKRMD n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Marker function-FCTN			
Marker mode	MKR n 0: NORMAL MKR 1: DELTA MKR 2: ZERO MKR	MKR?	MKR n
Functions using marker values	MKR n n=3 to 12	********************	********************
MKR→MAX	MKR 3		
MKR→MIN	MKR 4		
MKR CHANGE	MKR 5		
MKR→CF	MKR 6		
DELTA→SPAN	MKR 7		
MKR→OFS	MKR 8		
MKR→+PEAK	MKR 9		
MKR→-PEAK	MKR 10		
TRACK +PEAK	MKR 11		
TRACK -PEAK	MKR 12		
Marker tracking function	TRKMOD n 0: OFF 1: TRACK +PEAK 2: TRACK -PEAK	TRKMOD?	TRKMOD n
Marker list display	MKLST n 0: OFF/1:ON	MKLST?	MKLST n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Filter analysis function -FILTER			
Filter analysis function ON/OFF	FLTR SW, n n 0: OFF/1:ON	FLTR? SW	FLTR SW, n
Filter center frequency	FLTR CNF, f f: frequency	FLTR? CNF	FLTR CNF, f
Insertion loss calculation standard	FLTR TYP, n 0: FILTER CF 1: MAX VALUE	FLTR? TYP	FLTR TYP, n
Bandwidth calculation standard	FLTR BWR,n 0: FILTER CF 1: MAX VALUE	FLTR? BWR	FLTR BWR, n
X1 dB setting	FLTR X1D,r r= -200 to +200 (dB)	FLTR? X1D	FLTR X1D, r
X2 dB setting	FLTR X2D, r r= -200 to +200 (dB)	FLTR? X2D	FLTR X2D, r
Ripple search start frequency	FLTR STF,f f: frequency	FLTR? STF	FLTR STF, f
Ripple search end frequency	FLTR SOF, f f: frequency	FLTR? SOF	FLTR SOF, f
Ripple resolution	FLTR RSL, f f: frequency	FLTR? RSL	FLTR RSL, f
Frequency result display place number assignment	FLTR DGT, n n=3 to 8	FLTR? DGT	FLTR DGT, n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Filter analysis function -FILTER			
Calculation result read			
F0	*********************	FLTR? F0	frequncy
FL1	*********************	FLTR? FL1	frequncy
FR1	*********************	FLTR? FR1	frequncy
FL2	*********************	FLTR? FL2	frequncy
FR2	*********************	FLTR? FR2	frequncy
BW1	*********************	FLTR? BW1	frequncy
BW2	*********************	FLTR? BW2	frequncy
RIPPLE	*********************	FLTR? RPL	value
INSERTION LOSS	*********************	FLTR? IL	value
Q	*********************	FLTR? Q	value
SF	*********************	FLTR? SF	value
Total result read	*********************	RESF?	f0, bw1, fl1, fr1, bw2, fl2, fr2, il, rpl, q, sf

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Resonator analysis function -RESONATOR			
Resonator analysis function ON/OFF	RESO SW, n 0: OFF/1:ON	RESO? SW	RESO SW, n
Analysis method selection	RESO TYP, n 1: RESON1 2: RESON2	RESO? TYP	RESO TYP, n
Analysis band start frequency	RESO STF, f f: frequency	RESO? STF	RESO STF, f
Analysis band end frequency	RESO SOF, f f: frequency	RESO? SOF	RESO SOF, f
RESON1 analysis method	RESO FQP, n 0: ZERO PHASE 1: MIN/MAX IMPD	RESO? FQP	RESO FQP, n
Frequency result display place number assignment	RESO DGT, n n= 3 to 8	RESO? DGT	RESO DGT, n
Calculation result read			
Fs	********************	RESO? FS	frequency
Fr	********************	RESO? FR	frequency
Fa	********************	RESO? FA	frequency
Fn	********************	RESO? FN	frequency
Fm	********************	RESO? FM	frequency
Zr	********************	RESO? ZR	value
Za	********************	RESO? ZA	value
Zn	********************	RESO? ZN	value
Zm	********************	RESO? ZM	value
R1	********************	RESO? R1	value
C0	********************	RESO? C0	value
C1	********************	RESO? C1	value
L1	********************	RESO? L1	value
Q	********************	RESO? Q	value

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Resonator analysis function -RESONATOR			
Total result read			
RESON1-ZERO PHASE	********************	RESR?	fr, zr, fa, za
RESON1-MIN/MAX IMPD	********************	RESR?	fn, zn, fm, zm
RESON2	********************	RESR?	fr, zr, fa, za, fs, r1, l1, c0, c1, q

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Limit test function -LIMIT			
Limit test function ON/OFF	LIMT n 0: OFF/1:ON	LIMT?	LIMT n
Line type assignment	LMTP t, nX t:trace assignment 0: TRACE-A/B 1: TRACE-A 2: TRACE-B nX:line type 0: SILGLE 1: SEGMENTED	LMTP? 0 LMTP? 1 LMTP? 2	LMTP 1, nA; LMTP 2, nB LMTP 1, nA LMTP 2, nB
Single line - upper limit value setting	LSIU t, rX t:trace assignment 0: TRACE-A/B 1: TRACE-A 2: TRACE-B rX:Upper limit standard value	LSIU? 0 LSIU? 1 LSIU? 2	LSIU 1, rA; LSIU 2, rB LSIU 1, rA LSIU 2, rB
- Clear	LCLU t, 0 t: trace assignment	********************	********************
Single line - lower limit value setting	LSIL t, rX Same parameter settings as the single line upper limit value setting	LSIL? n n=0, 1, 2	Same response setting as the single line upper limit value setting
- Clear	LCLL t, 0 t: trace assignment	********************	********************

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Limit test function-LIMIT			
Segment line - upper limit value setting	LMTU t, m, f0, r0, f1, r1 t:trace assignment 0: TRACE-A/B 1: TRACE-A 2: TRACE-B m:segment number m=1 to 10 f0: start frequency r0: upper limit standard value for the start frequency f1: end frequency r1: upper limit standard value for the end frequency	LMTU? 0, m LMTU? 1, m LMTU? 2, m	LMTU 1, f0, r0, f1, r1; LMTU 2, f0, r0, f1, r1 LMTU 1, f0, r0, f1, r1 LMTU 2, f0, r0, f1, r1
- Clear	LCLU t, 1 t:trace assignment	********************	********************
Segment line - lower limit value setting	LMTL t, m, f0, r0, f1, r1 Same parameter settings as the segment line upper limit value settings	LMTL? n, m n=0, 1, 2	Same response setting as the segment line upper limit value setting
- Clear	LCLL t, 1 t:trace assignment	********************	********************
Limit test result read	********************	LMTS? t t: trace assignment 0: TRACE-A/B 1: TRACE-A 2: TRACE-B LMTS? 0 LMTS? 1 LMTS? 2	 LMTS 1, n; LMTS 2, n LMTS 1, n LMTS 2, n n= 0 : FAIL n= 1 : PASS n= 2 : No testing
Beep ON/OFF	BPON n 0: OFF/1ON	BPON?	BPON n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Hardcopy/copy control -COPY/COPY CONT			
Printer GPIB address setting	PRIA n n=0 to 30	PRIA?	PRIA n
Printer type	PRTM n n 0:ESC/P 1:HP 2:VIDEO OUT 3:FD (BMP)	PRTM?	PRTM n
Form feed ON/OFF	PFF n n 0:OFF/1:ON	PFF?	PFF n
Talker terminator switching	TRM n n 0:CR/LF+EOI 1:LF+EOI	TRM?	TRM n
Time-out setting (controller in use)	GTM r r=0 to 99999 (sec) Infinite wait if 0 sec is set	GTM?	GTM r
Hardcopy output port selection (active port selection)	PORT n n 1:GPIB 2:RS232C (Opt) 3:Centro (Opt)	PORT?	PORT n
Hardcopy start/stop and copy condition read	COPY n n 0:stop/1: start	COPY?	n 0:copy end (READY) 1:copying
Video plotter dedicated copy start	VPT	********************	********************
■ Hard copy in the BMP format			
Color selection	BMP COLOR, n 0: Monochrome 1:Color	BMP? COLOR	BMP COLOR, n
Color data compression	BMP COMP, n 0: OFF 1:ON	BMP? COMP	BMP COMP, n
Copy No.	BMP COPYNO, n n=0000 to 9999	BMP? COPYNO	BMP COPYNO, n

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ SAVE/RECALL			
Save item selection			
Measuring parameter	SV1 n	SV1?	SV1 n
S (normalize) data	SV2 n	SV2?	SV2 n
Trace data	SV3 n	SV3?	SV3 n
Frequency table	SV4 n	SV4?	SV4 n
Level table	SV5 n	SV5?	SV5 n
Calibration data	SV6 n 0: OFF 1: CH 1 and 2 selection 2: CH 1 selection 3: CH 2 selection	SV6?	SV6 n
RBW table	SVRBW m, n	SVRBW? m1	SVRBW m1, n
User wait table	SVWAI m, n m: channel selection 0: CH 1 and 2 selection 1: CH 1 selection 2: CH 2 selection n: selection ON/OFF 0: OFF/1:ON	SVWAI? m1 m1:channel designation 1:CH1 2:CH2	SVWAI m1, n
Drive selection	PMCS n 0: INT MEM 1: FD 2: PMC	PMCS?	PMCS n
Media format and directory creation	MA4	********************	********************
Directory creation	MKDR	********************	********************

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
Save execution	SVM r r=0 to 99	********************	********************
Recall execution	RCM r	********************	********************
Delete execution	DLM r	********************	********************
Index recall display	INDX n 0: CLOSE/1:OPEN	INDX?	INDX n
Save items list display/clear	RC2: display RC3: clear	******************** ********************	******************** ********************
Media error information read	********************	PER?	00:(NORMAL) 01:NO FD,PMC 02:NO FORMAT 04:PROTECTED 05:MEDIA ERROR 06:MEM. OVER 07:NOT FOUND 12:DIFFERENT VERSION 13:SELECT FDD 14:NO DATA 11:OTHER ERRORS
Save data in the text format	SVTXT r r=0 to 99	********************	********************

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Title -SAVE/RECALL			
Display ON/OFF	TTL n 0: OFF/1:ON	TTL?	TTL n
Title character input/read	TEN text	TEN?	TEN text
■ Clock -SYSTEM			
Year, month, and day setting/read	DATE yy, mm, dd yy: year (00 to 99) mm: month (1 to 12) dd: minute (1 to 31)	DATE?	DATE yy, mm, dd
Hour, minute, and second setting/read	TIME hh, mm, ss hh: hour (00 to 23) mm: minute (0 to 59) ss: second (0 to 59)	TIME?	TIME hh, mm, ss
■ Display color setting -SYSTEM			
Color setting for each screen	CPLL m, n m: Screen number (1 to 17) Note1 n: Color number (0 to 15) 0: pitch black 1: dark blue 2: dark red 3: dark purple 4: dark green 5: dark sky blue 6: dark yellow 7: light gray 8: dark gray 9: light blue 10: light red 11: light purple 12: light green 13: light sky blue 14: light yellow 15: pure white	CPLL? m	CPLL m, n

Note 1:

For the screen numbers, see the subsection “CFL subroutine” in Section 5.2 “System Subroutine” in the Remote Control (PTA Control) manual.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Back light -BACK LIGHT			
Back light ON/OFF	BLIGHT n 0: OFF/1:ON	BLIGHT?	BLIGHT n
■ PTA			
PTA ON/OFF	PTA n 0: OFF/1:ON	********************	********************
PTL mode (PTA program transfer mode)	PTL n 0: OFF(NORMAL) 1: ON (PC → measuring instrument) 2: ON (measuring instrument → PC)	********************	********************
PTA dual-port memory write/read	PMYm, d m: write block number (0 to 31) d: write data (any numerics and characters) or PMY m1 d1 d2 • • m1: write start block number (0 to 31) dX: write data (any numerics and characters)	PMY? m1, n m1: read start block number (0 to 31) n: number of read blocks (1 to 32)	d1 d2 • • dn

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Trace data/normalize data read format			
Read format assignment 1 (ASCII format/BINARY format)	BIN n 0: ASCII format 1: BINARY format	BIN?	BIN n
Read format assignment 2 (Floating-point type/fixed-point type)	FRMT n 0: floating-point type 1: fixed-point type Valid if the read format 1 is ASCII format	FRMT?	FRMT n
■ Access channel selection			
Read/write of active channel side	SRW ACT	SRW?	SRW ACT
Read/write of CH 1 side	SRW CH1	SRW?	SRW CH1
Read/write of CH 2 side	SRW CH2	SRW?	SRW CH2
■ Data table protect Note 1	TBGRD n 0: OFF / 1:ON	TBGRD?	TBGRD n

Note 1:

Device message used to protect the frequency, level, RBW, and user wait tables from being updated due to parameter settings.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Trace data read and write			
TRACE-A	XMA m, n, d1, d2••, dn or XMA m, n (XMAD) d1 (XMAD) d2 • • (XMAD) dn m: write start point (0 to 1000) n: number of write points (1 to 1001) dX: trace data (-838.8608 to +838.8607) (XMAD) can be omitted	XMA? m, n m:read start point n:number of read data	d1 d2 • • dn
TRACE-B	XMB m, n, d1, d2••, dn or XMB m, n (XMBD) d1 (XMBD) d2 • • (XMBD) dn m, n, and dX are the same as for TRACE-A (XMBD) can be omitted	XMB? m, n m and n are the same as for TRACE-A	Response data is the same as for TRACE-A

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Normalize data (S memory) read and write			
S memory on TRACE-A side	SMA m, n, d1, d2, ••, dn or SMA m, n (SMAD) d1 (SMAD) d2 • • (SMAD) dn m, n, and dX are the same as for TRACE-A (SMAD) can be omitted	SMA? m, n m and n are the same as for TRACE-A	Response data is the same as for TRACE-A
S memory on TRACE-B side	SMB m, n, d1, d2, ••, dn or SMB m, n (SMBD) d1 (SMBD) d2 • • (SMBD) dn m, n, and dX are the same as for TRACE-A (SMBD) can be omitted	SMB? m, n	Response data is the same as for TRACE-A
■ Reading of the complex measurement data			
Reading of the real number section Reading of the imaginary number section	******************** ********************	CDR? m, n CDI? m, n m and n are the same as for TRACE-A	Response data is the same as for TRACE-A. However, only floating-point type is used for the ASCII format. Only real number type is used for the BINARY format

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Frequency table data read and write			
Assigned channel side Note 1 **Note 1:** Writing or reading of the access channel side assigned with "SRW" command	FQM m, n, d1, d2, ••, dn or (FQMD) f1 (FQMD) f2 • (FQMD) fn m and n are the same as for TRACE-A fX: frequency data 0 to 300 000 000(Hz) (FQMD) can be omitted	FQM? m, n m and n are the same as for TRACE-A	f1 f2 • • fn Only the ASCII format and fixed-point type
CH 1 side	FQMA m, n, d1, d2, ••, dn or FQMA m, n (FQMAD) f1 (FQMAD) f2 • (FQMAD) fn (FQMAD) can be omitted	FQMA? m, n m and n are the same as for TRACE-A	f1 f2 • • fn Only the ASCII format and fixed-point type
CH 2 side	FQMB m, n, d1, d2, ••, dn or FQMB m, n (FQMBD) f1 (FQMBD) f2 • (FQMBD) fn (FQMBD) can be omitted	FQMB? m, n	f1 f2 • • fn Only the ASCII format and fixed-point type

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Level table data (used for power sweep) read and write			
Assigned channel side Note 1 **Note 1:** Writing or reading of the access channel side assigned with "SRW" command	LVM m, n, d1, d2, ••, dn or LVM m, n (LVMD) d1 (LVMD) d2 • (LVMD) dn m and n are the same as for TRACE-A dX: Level data Same setting rage as the output power (LVMD) can be omitted	LVM? m, n m and n are the same as for TRACE-A	d1 d2 • • dn Only the ASCII format and fixed-point type
CH 1 side	LVMA m, n, d1, d2, ••, dn or LVMA m, n (LVMAD) d1 (LVMAD) d2 • (LVMAD) dn (LVMAD) can be omitted	LVMA? m, n m and n are the same as for TRACE-A	d1 d2 • • dn Only the ASCII format and fixed-point type
CH 2 side	LVMB m, n, d1, d2, ••, dn or LVMB m, n (LVMBD) d1 (LVMBD) d2 • (LVMBD) dn (LVMBD) can be omitted	LVMB? m, n m and n are the same as for TRACE-A	d1 d2 • • dn Only the ASCII format and fixed-point type

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ RBW table data read and write Assigned channel side Note 1	RBTB m, n, d1, d2, ••, dn or RBTB m, n (RBTBD) d1 (RBTBD) d2 • (RBTBD) dn m and n are the same as for TRACE-A dX:RBW code Same as setting code for RBW parameter (RBTBD) can be omitted	RBTB? m, n m and n are the same as for TRACE-A	d1 d2 • • dn
■ User wait table data read and write Assigned channel side Note 1 **Note 1:** Writing or reading of the access channel side assigned with "SRW" command	UWTB m, n, d1, d2, ••, dn or UWTB m, n (UWTBD) d1 (UWTBD) d2 • (UWTBD) dn m and n are the same as for TRACE-A dX:Wait time 0.01 to 7200000 (msec) (UWTBD) can be omitted	UWTB? m, n m and n are the same as for TRACE-A	d1 d2 • • dn

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Common commands			
Rset Command	*RST	********************	********************
Identification Query	********************	*IDN?	ANRITSU, MS4630B, 0,0000
SelfTest Query	********************	*TST?	n:-32768 to 32767 Definition of each bit in binary notation is given below. 0: MAIN CPU 1: DISP CPU 2: DSP 3: (RESERVED) 4: LOCAL 10M PLL 5: 1st LOCAL 6: (RESERVED) 7: (RESERVED) 8: OUT 100M PLL 9: OUT 50M PLL 10: OUT ALC 11: (RESERVED) 12: REC. (Rch) 13: REC. (TAch) 14: REC. (TBch) 15: (RESERVED) Each bit is 0: OK/1: NG
Operation Complete	*OPC	*OPC?	n (=1)
Wait to Continue Command	*WAI	********************	********************
Clear Status Command	*CLS	********************	********************
Read Status Byte Query	********************	*STB?	n:0 to 255 Bit0:unused 1: PTA 2: END 3: ERROR 4: MAV 5: ESB 6: RQS&MSS 7: unused

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Common commands			
Service Request Enable	*SRE n n: 0 to 255	*SRE?	n:0 to 255
Standard Event Status Regster	********************	*ESR?	n: 0 to 255 Bit 0:OPC 1: unused 2: QYE 3: unused 4: EXE 5: CME 6: unused 7: PON
Standard Event Status Enable	*ESE n n: 0 to 255	*ESE?	n:0 to 255
Power On Status Clear	*PSC	*PSC?	n (=0)
Trigger Command	*TRG Single sweep execution	********************	********************
Recall Command	*RCL n n=0 to 99 Same as SVM n	********************	********************
Save Command	*SAV n n=0 to 99 Same as RCM n	********************	********************

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Common commands Option Identification Command	********************	*OPT?	n:-32768 to 32767 Definition of each bit in binary notation is given below. 0:REC.(TBch) 1:ATT(ELEC.) 2:ATT(MECH.) 3:3WAY DIVIDER 4:REF.OSC 5:PMC DRIVE 6:RS232C/Centro. 7:75Ohm Adapter 8 to 15:(RESERVED) Common to all bits 0:none/1: present

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ GPIB dedicated command			
Event Status register 1 (PTA Status register)	********************	ESR1?	n:0 to 255 Bit0 to 3:CALL RSV Bit4 to 7:PTA ERR.
Enable register 1	ESE1 n n: 0 to 255	ESE1?	n
Event Status register 2 (End Status register)	********************	ESR2?	n:0 to 255 Bit0: CH 1 sweep 1: CH 2 sweep 2: FD drive 3: utility function 4 to 7: (RESERVED)
Enable register 2	ESE2 n n: 0 to 255	ESE2?	n
Event Status register 3 (System Error Status register)	********************	ESR3?	n:0 to 255 Bit0: FAN stops 4 to 7: (RESERVED)
Enable register 3	ESE3 n n: 0 to 255	ESE3?	n:0 to 255

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Drawing-related command			
PTA drawing origin moving Note 1	GORG x, y x: -639 to 639; origin X-coordinate y: -479 to 479; origin Y-coordinate Note 1	GORG?	GORG x, y
Screen deletion	CFL n n:screen number Note 2	*********************	*********************
Character drawing	DCH x, y, text, m (,n) x, y: drawing coordinates m: screen number n: normal (0)/reverse (1) Note 2	*********************	*********************
Line drawing	DLN x0, y0, x1, y1, m (,n) x0, y0:starting point coordinates x1, y1:ending point coordinates m: screen number n: line type Note 2	*********************	*********************
Quadrilateral drawing	DRC x0, y0, x1, y1, m (,n) x0, y0:Coordinates of opposite angle 0 x1, y1: Coordinates of opposite angle 1 m: screen number n: line type Note 2	*********************	*********************

Note 1:

The move of drawing coordinates is applicable only when the system subroutine function of PTA is used for drawing. It is not applicable to a device message due to this remote control.

Note 2:

For details of parameters of drawing commands, see the subsection "system subroutines" in Section 5 "Extended PTL" in the Remote Control (PTA Control) manual.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Drawing-related command			
Circle drawing	DCR x, y, r, m (,n) x, y: circle center coordinates r: circle radius m: screen number n: line type Note	********************	********************
Arc drawing	DAR x, y, r, s1, s2, m (,n) x, y: circle center coordinates r: circle radius s1: starting angle s2: ending angle m: screen number n: line type Note		
○ and × drawing	DJG x, y, f, m1, m2 x, y: circle center coordinates r: circle radius f: function ○ display (0)/× display (1) m1: ○ drawing screen number m2: × drawing screen number Note		

Note:

For details of parameters of drawing commands, see the subsection “system subroutines” in Section 5 “Extended PTL” in the Remote Control (PTA Control) manual.

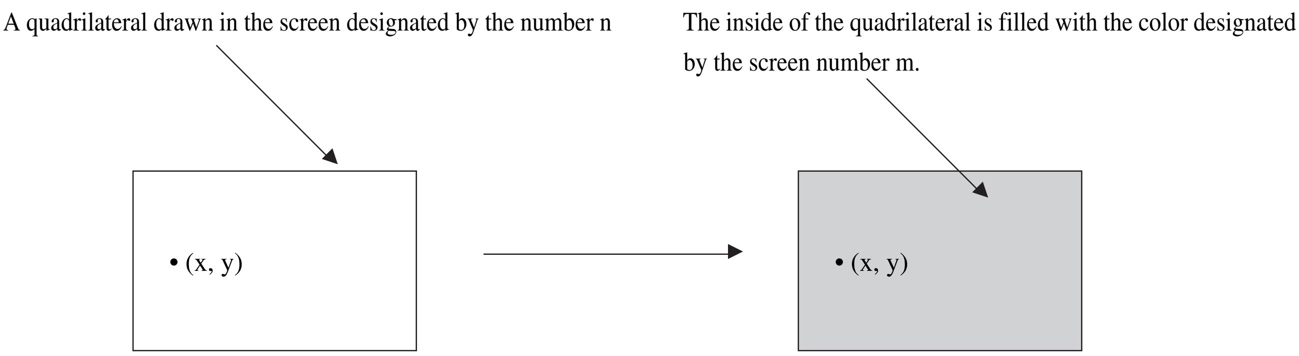
MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Drawing-related command Color fill drawing	PNT x, y, m, n x, y: assigned coordinates m: screen number for color fill (1 to 17) n: screen number for area assignment (0 to 17)	*********************	*********************

Using the color fill drawing command:

If a point designated by (x, y) coordinates exists within a circle or quadrilateral (a closed area with matching starting and ending points of drawing) drawn in the screen designated by the number n, this command fills the inside of this area with the color designated by the screen number m.

For details of screen numbers, see the subsection “system subroutines” in Section 5 “Extended PTL” in the Remote Control (PTA Control) manual.



Note:

If a circle or quadrilateral (a closed area with matching starting and ending points of drawing) drawn in the screen designated by the number n shares intersection point(s) with a line or quadrilateral drawn in a screen other than that designated by the number n, the former is considered to have non-matching starting and ending points of drawing, and thus the areas other than the closed area will also be filled with color.

MS4630B List of device messages (Continued)

Function summary	Control command	Query	Response
■ Arithmetic function Function of calculating the group delay time from complex data	********************	GPDLY? P0, P1, ch [,e] P0, P1: measuring point (0 to 1000) ch: measurement channel (1: CH-1/2: CH-2) e: decimal place assignment (0 to -12)	t:(Group delay result)

Note:

For information on using the group delay function, see Section 9, “Other Device Messages”.

Section 7
Reading Sweep Control/Measurement Data

7.1 Overview

If measurement data is read after a sweep is started from an external controller and the sweep end is detected, special commands and procedures are required which are not found in manual operations.
This chapter explains about the sweep start commands, sweep end detection, and procedure for reading measurement data.

7.2 Sweep Start Commands and Sweep Condition (Sweep End) Detection

7.2.1 Sweep start

Use one of the following remote commands to start a single sweep.

Remote command	Function
"SWP 1"	Only starts a normal single sweep (without sweep end synchronization).
"SWP 2"	Starts and ends a normal single sweep (with sweep end synchronization).
"CAL"	Only starts the X → S sweep (normalize sweep) (without sweep end synchronization).
"CAL 2"	Starts and ends the X → S sweep (normalize sweep) (with sweep end synchronization).
"COR n"	Starts a calibration data uptake sweep. n varies depending on the calibration data type.

Notes:

- If any of the above sweep start remote commands is executed while executing a sweep, the sweep is restarted.
- In the case of a sweep using an external trigger, the sweep is started only after any of the above sweep start remote commands is executed and then an external trigger event occurs.
- If the measurement channels are channels 1 & 2 and the coupled channel is OFF, each channel is swept alternately. The single sweep in this case is sweeping each of the channels once with the active channel as the preceding sweep channel.

7.2.2 Sweep condition (sweep end) detection

Use one of the following remote commands to query the sweep condition.

Remote command	Response data
"SWP?"	"n" 0: sweep end/(1: sweeping)
"CAL?"	"n" 0: sweep end/(1: sweeping)
"COR?"	"n" 0: sweep end/(1: sweeping)

The response data (0: sweep end) is returned when the sweep ends or halts.
If the sweep condition is queried while a sweep is executed, no response is returned, and the controller is forced to wait.
That is, no return of responses mean that a sweep is being executed.

Therefore, if the sweep condition is queried after sending a sweep start remote command, the time when its response data is received on the controller side can be recognized as the sweep end point.

If the "SWP 2" or "CAL 2" command is used to execute a sweep, the command covers the range between the start and end of the sweep. In this case, even if the above sweep condition is not queried, the time when the command execution ends can be recognized as the sweep end point.
That is, this command is useful in an application program that executes a next command as soon as the sweep ends.

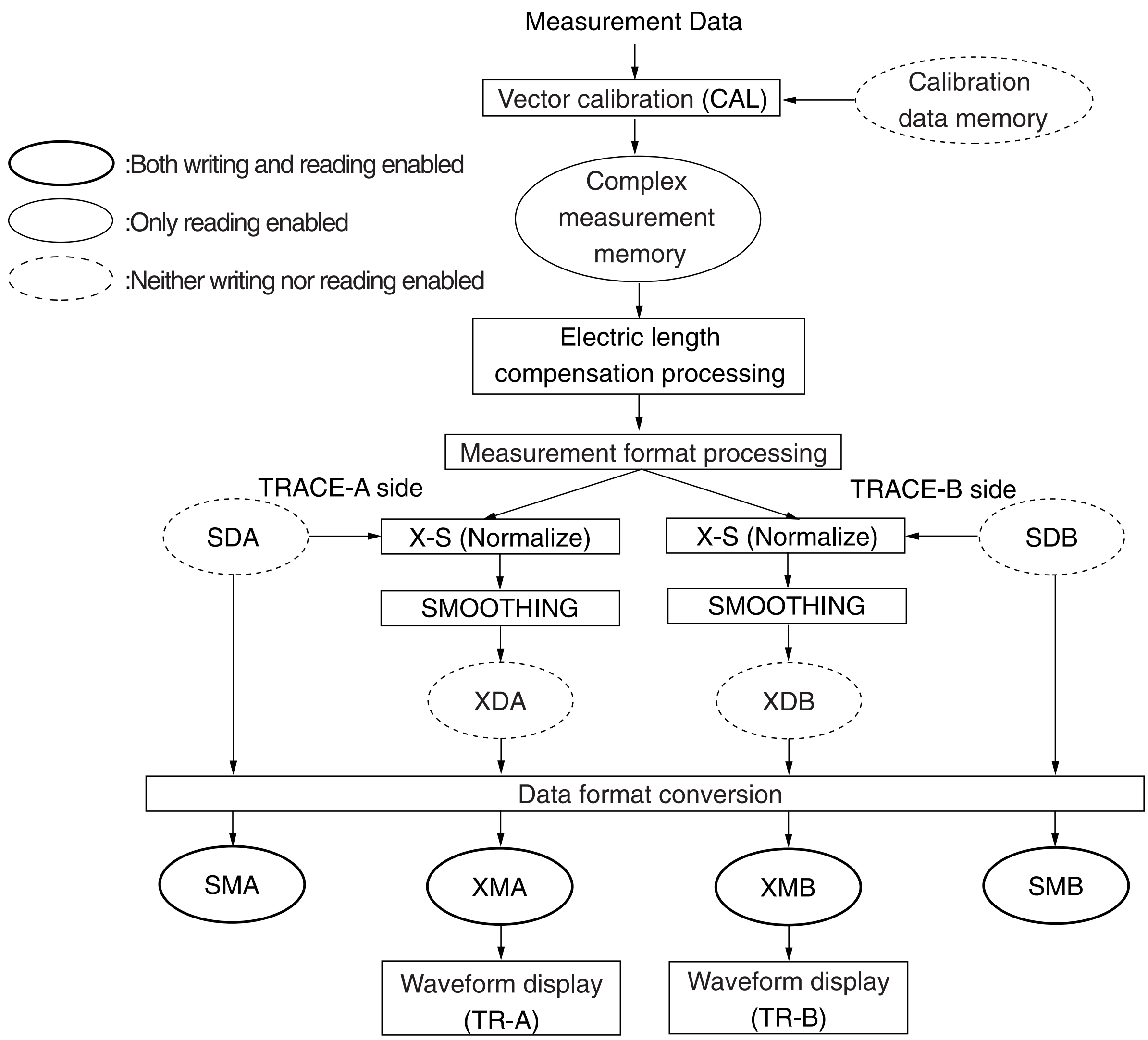
Note:

While the controller waits for response data, or while a command with sweep end synchronization is executed, press the LOCAL key on the front panel of the measuring instrument to cancel the wait state or the command execution state.

7.3 Reading Measurement Data

7.3.1 Measurement memory/Waveform memory

The sweep allows you to output measurement data (waveform data) stored inside measuring instruments to an external controller. The following shows the configuration, application and attribute of the measurement memory and waveform memory to be read from in this case.



Measurement Memory/Waveform Memory Calibration Flow

Note:

The configuration shown in the figure above illustrates a single measurement channel.
As the device has two measurement channels (ch1 and ch2), the measurement memory and waveform memory shown in the figure above exist for each measurement channel.

Measurement Memory/Waveform Memory Application

Memory name	Application
Calibration data memory	Complex data memory used to store the calibration data obtained through vector calibration (CAL).
Complex measurement memory	Complex data memory used to store the measurement data that has been vector-calibrated. However, the X-S (normalization), smoothing process or subtrace process has not been performed on the measurement data.
XDA	Measurement memory on the TRACE-A side used to store, in the internal format of the measuring instrument, the measurement data that has been formatted.
XDB	Measurement memory on the TRACE-B side used for the same purpose as the above.
SDA	Measurement memory on the TRACE-A side used to store, in the internal format of the measuring instrument, the reference data (S data) to be subject to the X-S (normalization) function.
SDB	Measurement memory on the TRACE-B side used for the same purpose as the above.
XMA	Waveform memory on the TRACE-A side used to store the XDA data converted to a format for display and external input-output.
XMB	Waveform memory on the TRACE-B side used for the same purpose as the above.
SMA	Waveform memory on the TRACE-A side used to store the SDA data converted to a format for external input-output.
SMB	Waveform memory on the TRACE-B side used for the same purpose as the above.

Notes:

- The number of data arrays in each measurement memory and waveform data memory is 1001 points.

- When the measurement format is either POLAR, IMPD CHART or ADMT CHART, the value of the real section within the complex data is stored in measurement memory A (XDA, SDA, XMA, SMA) and the value of the imaginary section within the complex data is stored in measurement memory B (XDB,SDB, XMB, SMB).

  When the measurement format is LOG Z, the following relationship is established between the data (XMA) and the measured value within the measurement memory (Z: impedance).

  $Z\ (\Omega) = 10^{(XMA/10)}/1000$

Measurement Memory/Waveform Memory Attribute

Memory name	Writing/ Reading	Reading of the ASCII format		Reading of the BINARY format		Saving to a file
		Fixed-point	Floating-point	Integer	Real number	
Calibration data memory	---	---	---	---	---	○
Complex measurement memory	RD	---	○	---	○	---
XDA, XDB	---	---	---	---	---	○
SDA, SDB	---	---	---	---	---	---
XMA, XMB	WR/RD	○	○	○	---	---
SMA, SMB	WR/RD	○	○	○	---	---

7.3.2 Read data format

The types of data to be read from the measurement memory or the waveform memory are roughly classified into two types: the ASCII (character string) and BINARY formats. The ASCII (character string) format is further divided into the fixed-point and floating-point types while the BINARY format into the real number and integer types.
Select one of these data formats depending on whether the measurement memory or waveform memory is used as follows:

Output in the ASCII (character string) format	Floating-point type	: XMA, XMB, SMA, SMB, Complex measurement memory
	Fixed-point type	: XMA, XMB, SMA, SMB
Output in the BINARY format	Real number type	: Complex measurement memory
	Integer type	: XMA, XMB, SMA, SMB

7.3.3 Reading data in the ASCII (character string) format

In the ASCII (character string) format, one digit of measurement data (including the decimal point, sign, and exponential symbol) is output as one-byte ASCII character.

One of the two types can be selected for the ASCII (character string) format: the floating-point type in the format of mantissa + exponent with a varying decimal point depending on data or the fixed-point type with a fixed decimal point used to represent, e.g., a marker level.

Advantages and disadvantages of reading data in the ASCII (character string) format

Advantages:

- Since data is sent to and received from the controller in the same format as for normal remote commands, a program can be easily developed for the controller, and the data processing after reading is generally easy.

Disadvantages:

- In general, the size of data transferred from the measuring instrument is larger than that in the BINARY format, requiring more time for data transfer and reading by the controller.

(1) ASCII (character string): Floating-point type output

- The measurement data taken at each point consists of a mantissa and an exponent when it is output.
  Additionally, the data is output as a real number allowing for a measurement unit according to the measurement format used when the data is taken.

Overall sign/Mantissa/E/Exponential sign/Exponential value

Overall sign	: + or - is attached. + is not omitted even if the value is positive.
Mantissa	: Real number consisting of a 1-digit integer part, decimal point, and 6 decimal places.
E	: The numeric value following this symbol is an exponential part.
Exponential sign	: + or - is attached. + is not omitted even if the value is positive.
Exponential value	: 2-digit integer value.

Output example –1.234000E–01

If the measurement format is LOGMAG, the above output value represents a measured value of $-1.234*10^{-1}$ dB = -0.1234 dB.

- The measurement memory and the waveform memory that can output data in the ASCII (character string) format: floating-point type are XMA, XMB, SMA, SMB, and complex measurement memory.

(2) ASCII (character string): Fixed-point type output

- All measurement data taken at each point is output in the same format as the display format of the numeric part of the marker level.
  Additionally, the data is output as a real number allowing for a measurement unit according to the measurement format used when the data is taken.

Output example –12.3456

If the measurement format is LOGMAG, the above output value represents a measured value of –12.3456 dB.

- The measurement memory and the waveform memory that can output data in the ASCII (character string) format: fixed-point type are XMA, XMB, SMA, and SMB.

7.3.4 Procedure for reading measurement data - ASCII (character string) format

The following shows an example of reading measurement data (0 to 250 points) into the controller from the waveform memory on the TRACE-A side of the measurement channel 1 (XMA).

"BIN 0" : Specification of reading data in the ASCII (character string) format (0: ASCII/1: BINARY)

↓

"FRMT 1" : Specification of "fixed-point type" in the ASCII (character string) format (0: floating-point/1: fixed-point)

↓

"SRW CH1" : Specification of reading data from Channel 1 (ACT: active channel side/CH1: channel 1 side/CH2: channel 2 side)

↓

"XMA? 0,251" : Request for outputting 251 pieces of data from the TRACE-A waveform memory (XMA), i.e., data from the 0th to 250th point.

↓

Reading and storing data at one point → → X(n)   Looping for 251 times

↓

End

-Order of output data-

Data at 0th point	Delimiter
Data at 1st point	Delimiter
Data at 2nd point	Delimiter
·	
·	
Data at 250th point	Delimiter

Note:

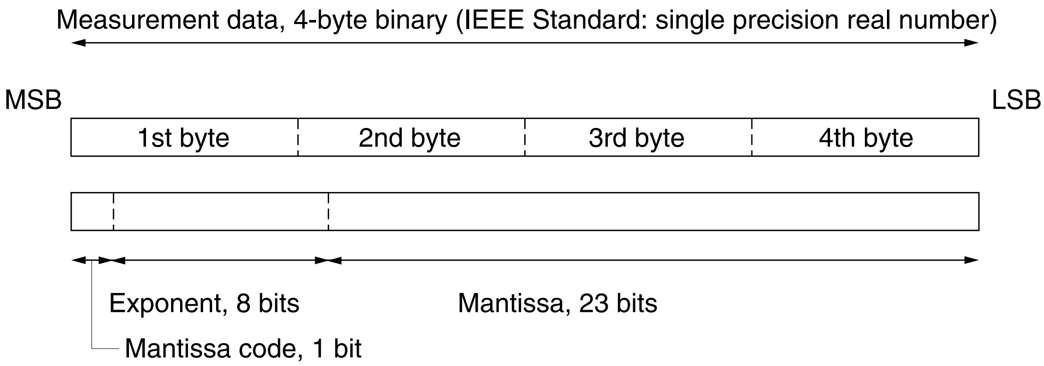
After each piece of data is output, CR+LF (2 bytes) or LF (1 byte) is output as a delimiter. The delimiter may be changed using the "TRM n" command.

7.3.5 Reading data in the BINARY (binary) format

In the BINARY (binary) format, measurement data is output as binary values with a fixed 4-byte length.
Advantages and disadvantages of reading data in the BINARY format can be generally assumed as being opposite to those of reading data in the ASCII (character string) format.
As with the ASCII (character string) format, the BINARY format is divided into two types: real number type and integer type. The data is output in either the real number or the integer type depending on the measurement memory or the waveform memory that outputs the data.

(1) BINARY: real number output

Output format of BINARY: real number type data



- All measurement data taken at each point is output as four-byte (32-bit) data.
- If the 1st byte is on the MSB side and the 4th byte is on the LSB side, the measuring instrument outputs 1st, 2nd, 3rd, and 4th bytes in this order.
- Only the complex measuring memory can output data in the BINARY format: real number type. The unit of real number and imaginary number data to be output in this case is an absolute number.

(2) BINARY: integer type output

Output format of BINARY: integer type data

Measurement data, 4-byte binary

MSB | 1st byte | 2nd byte | 3rd byte | 4th byte | LSB

- All measurement data taken at each point is output as four-byte (32-bit) integer data.

  The resolution of measurement data is determined by the measurement format used when the data is taken. That is, the place of the minimum resolution of measurement data at the time of measurement is the 10^0 place for this integer data.
  For example, if LOGMAG is used as the measurement format, 0.0001 dB is output as 1.

- As a negative number, the entire number (4-byte) is represented as a complement of 2.

- If the 1st byte is on the MSB side and the 4th byte is on the LSB side, the measuring instrument outputs the 1st, 2nd, 3rd, and 4th bytes in this order.

  <u>In this case, the controller must store each byte data that it has read not as character variables but as unsigned byte data of numerical variables that permits mathematical operations.</u>

- The following shows the operation in which the controller reverts each byte data that it has read to the original measurement data.

Numerical variables in the controller

1st byte data : A
2nd byte data : B
3rd byte data : C
4th byte data : D

→ $RES=A*2^{24}+B*2^{16}+C*2^{8}+D*2^{0}$

The operation is performed in view of the overall sign.

If $0 \le A \le 127$, that is, the data as a whole is positive or 0,
<u>Measured data = RES</u>
If $128 \le A \le 255$, that is, the data as a whole is negative,
<u>Measured data = RES–2^{32}</u>

- The measurement memory and the waveform memory that can output data in the BINARY format: integer type are XMA, XMB, SMA, and SMB.

7.3.6 Procedure for reading measurement data - BINARY format

The following shows an example of reading measurement data (0 to 250 points) into the controller from the waveform memory on the TRACE-A side of the measurement channel 1 (XMA).

"BIN 1" : Specification of reading data in the BINARY (binary) format (0: ASCII/1: BINARY)

↓

"SRW CH1" : Specification of reading data from Channel 1
(ACT: active channel/CH1: channel 1/CH2: channel 2)

↓

"XMA? 0,251" : Request for outputting 251 pieces of data from the TRACE-A waveform memory (XMA), i.e., data from the 0th to 250th point.

↓

(Setting the controller to 1-byte binary data input mode)

↓

Reading one byte (1st byte) → Storing it in numerical variable A

↓

Reading one byte (2nd byte) → Storing it in numerical variable B

↓

Reading one byte (3rd byte) → Storing it in numerical variable C

↓

Reading one byte (4th byte) → Storing it in numerical variable D

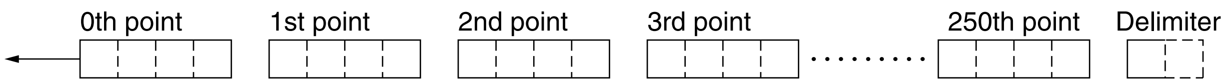
↓

Calculating the measured value from A, B, C, and D and storing it → X(n)

Looping for 251 times

↓

Reading one or two bytes (Reading the delimiter and discarding it)



Note:

After the final piece of data is output, CR+LF (2 bytes) or LF (1 byte) is output as a delimiter. The delimiter may be changed using the “TRM n” command.

Section 8
Writing and Reading Measurement Condition Data Table

8.1 Overview of Measurement Condition Data Table

The device stores in itself the measurement conditions corresponding to the measurement points, as data tables. In general, setting the measurement parameters automatically creates these tables. To change the condition data corresponding to a given measurement point in these tables, use a remote command or operate the front panel. The changed data tables can be saved to or recalled from auxiliary storages such as floppy disks, internal memory, and memory cards.

For details of data tables for measurement conditions, see Section 5.4.6, “List sweep” in the Vol. 1 Panel Operation manual.

Note:

Setting a measurement parameter automatically creates these data tables. Even though this function is used to change any given data in data tables, setting a measurement parameter again updates the contents of these data tables.
Therefore, a function is provided to prevent the data tables from being updated when a measurement parameter is set (the table guard function).

“TBGRD n”   0: table guard OFF (initial state)
            1: table guard ON

8.2 Device Messages for Writing and Reading

The following shows the device messages used to write and read data to measurement condition data tables.

Data table	Command header name	Measurement channel for writing and reading
Frequency data table	FQM	Access channel Note
	FQMA	Measurement channel 1
	FQMB	Measurement channel 2
Level data table	LVM	Access channel Note
	LVMA	Measurement channel 1
	LVMB	Measurement channel 2
RBW data table	RBTB	Access channel Note
User wait data table	UWTB	Access channel Note

Note:

The “SRW” command allows you to specify the access channel.

“SRW ACT”: Specifies the active channel as the access channel (initial state).

“SRW CH1”: Specifies the measurement channel 1 as the access channel.

“SRW CH2”: Specifies the measurement channel 2 as the access channel.

Frequency data table

- The data is input and output in real numbers, assuming 1 Hz=1.
- The setting range is 0 Hz to 300,000,000 Hz.
- The data is input and output in the ASCII (character string) format.

Level data table

- The data is input and output in real numbers, assuming 1 dBm=1.
- The setting range is the same as for the OUTPUT POWER setting parameter.
- The data is input and output in the ASCII (character string) format.

RBW data table

- As when the RBW is set as a measurement parameter, the data is input and output in the numerical code corresponding to each RBW.
- The data is input and output in the ASCII (character string) format.

User wait data table

- The data is input and output in real numbers, assuming 1 msec=1.
- The setting range is from 0.01 m sec to 7,200,000 m sec.
- The data is input and output in the ASCII (character string) format.

Note:

The number of data arrays in all the measurement condition data tables is 1001 points.

8.3 Procedure for Writing Data to Data Tables

The following shows an example of writing data to a frequency data table.
The procedure of writing data to other data tables is the same as below.

Format of writing data to a frequency data table :

"FQM m, n, d1, d2, ...dn"

m: Sets the measurement point at which to start writing data (0 to 1000).
The leftmost measurement point in the measurement screen is 0, and the rightmost point is (the number of measurement points - 1).

n: Sets the total number of measurement points at which to write data (1 to 1001).
The frequency data from the measurement point, m to m+n-1 is re-written.

d1, d2, ...dn: Sets the frequency data to write.
Write as many data pieces as n, separating them with commas.

Example of writing:

"FQM 0, 3, 10MHZ, 20MHZ, 30MHZ"

The frequency data of 10 MHz, 20 MHz, and 30 MHz for three points starting from Point 0, i.e., at Points 0, 1, and 2 is written to the frequency tables for designated measurement channels.

Alternately, you may use the following format if there are many data points to write data to:

"FQM m, n"
"FQMD d1"
"FQMD d2"
•
•
"FQMD dn"

Note:

To send only write data from the controller to the measuring instrument, attach a header indicating data at the beginning of each send data (add "D" at the end of a command header requesting for writing). This data header may be omitted.

Example:

"FQMD 1230000" : Meaning that the designated channel is written to at the frequency of 1,230,000 Hz.
"FQMBD 456000" : Meaning that Channel 1 is written to at the frequency of 456,000 Hz.

8.4 Procedure for Reading Data from Data Tables

The following shows an example of reading data from a frequency data table.
The procedure of reading data from other data tables is the same as below.

Format of reading data from a frequency data table :

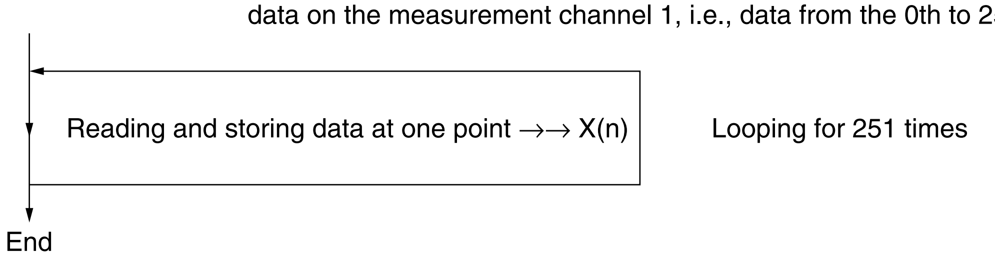
"FQM? m, n"

m: Sets the measurement point at which to start reading data (0 to 1000).
The leftmost measurement point in the measurement screen is 0, and the rightmost point is (the number of measurement points - 1).

n: Sets the total number of measurement points at which to read data(1 to 1001).
The frequency data from the measurement point, m to m+n-1 is read.

The following shows an example of reading data for 251 points in total (Points 0 to 250) from the beginning of the frequency data table on the measurement channel 1 to the controller.

"FQMA? 0,251" : Request for outputting 251 pieces of data from the frequency data/level data on the measurement channel 1, i.e., data from the 0th to 250th point.



-Order of output data-

Data at 0th point	Delimiter
Data at 1st point	Delimiter
Data at 2nd point	Delimiter
•	
•	
Data at 250th point	Delimiter

The delimiter is either CR+LF or LF.

Section 9
Details of Other Device Messages

9.1 Group Delay Measurement Function

9.1.1 Format of Device Message

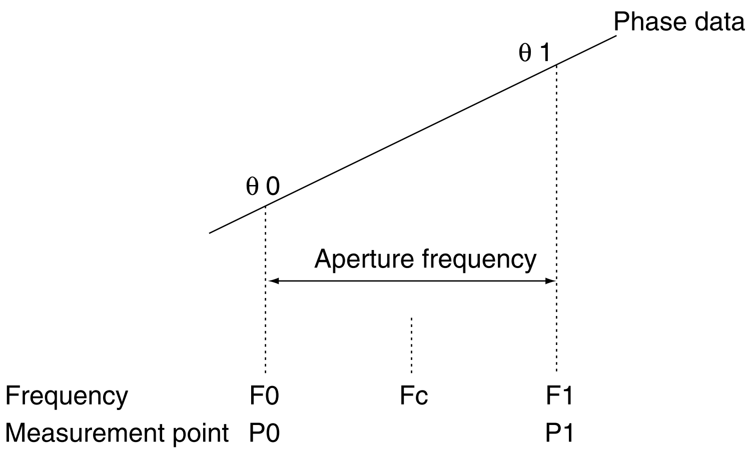
Inquiry message: "GPDLY? P0, P1, CH, [, E]"
Response data: t (group delay time)

P0 :Specification of measurement point at a lower frequency (0 to 1000)
P1 :Specification of measurement point at a higher frequency (0 to 1000)
CH :Specification of measurement channel (CH1: 1/CH2: 2)
E :Specification of decimal place in calculation result (0 to -12)

9.1.2 Calculating the group delay time

Using the measurement data (magnitudes of vector) and measurement frequencies at two measurement points, calculate a phase difference and a frequency difference to obtain the group delay time.
The frequency difference in this case is called the aperture frequency.

Obtaining the group delay time at the frequency, Fc (center frequency between F0 and F1)



Procedure of calculation

1. Use the measurement and frequency data for the measurement channel designated by CH.
2. Read the measured values (magnitude of vector) at the measurement points designated by P0 and P1.
3. Convert each of the measured values to a magnitude of phase. $\rightarrow \theta\,0, \theta\,1$ (deg)
4. Read the measurement frequencies at the measurement points designated by P0 and P1. $\rightarrow$ f0, f1 (Hz)
5. The value obtained in the following calculation is defined as the group delay time.
   $-(\theta\,1 - \theta\,0) / \{360 \times (f1 - f0)\} \rightarrow \tau$
6. Using the value designated by E, perform the following calculation and store the result in a numerical variable.
   $\tau \times 10^{E} \rightarrow$ Stored in a numerical variable.
   The assignment of E may be omitted. In this case, E=0 is assumed for calculation.

Note:

- As the measurement data to be converted into magnitudes of phase, use the vector data in the complex measurement memory. Since this data is not yet processed according to the measurement format such as LOGMAG or PHASE, the result of this function does not depend on the measurement format. However, the measurement data compensation or manipulation due to the X-S, smoothing, and subtrace processing performed after the data processing according to the measurement format is not reflected in the result of this function.
- If the frequency difference (f1 - f0) is 0 Hz, 0 is returned as the function result.
- If the value of (θ1 - θ0) is larger than 180 deg, or smaller than -180 deg, subtract or add 360 deg from/to it respectively and assume the result as the phase difference.

9.1.3 Use of this function

This function allows you to calculate the group delay time at a given frequency point at a given aperture frequency.
Using this function, you can obtain the group delay time values only for some predetermined frequencies, regardless of the format.
The following precautions apply for using this function.

- The data compensated or manipulated using the normalize (X-S), smoothing, or subtrace function (scaler data) cannot be used.
  Therefore, the compensation or manipulation of data normally performed in a measuring instrument must be performed in an external application program of the user.
- The data at two measuring points is needed to obtain the group delay time at one measurement frequency.

The following shows an example of calculating the group delay time using this function including the use of normalize (X-S) calibration.

1. Obtain the group delay time at the frequency, Fc and the aperture frequency, Fap.
2. Obtain the frequencies at the two measurement points as follows:
   F0=Fc-Fap/2 (the lower frequency)
   F1=Fc+Fap/2 (the higher frequency)
3. In the frequency data table for the channel to be measured, set F0 and F1 at measurement points, P0 and P1, respectively (P0 < P1). As required, set data also in the user wait table and RBW table.
   For information on setting data in the frequency, user wait, and RBW data tables, see Section 8, "Writing and Reading Data to and from Measurement Condition Data Table."
4. Directly connect the input and output of this measuring instrument to perform the normalize calibration sweep (X → S, X-S).
5. After the calibration sweep is completed, execute this group delay function to read the response data → τ s (τ s represents the group delay time before the calibration).
6. Connect the unit under measurement to perform sweep measurement.
7. After the sweep is completed, execute this function again to read the response data. → τ x
8. Obtain the group delay time after the normalize calibration at the frequency, Fc as: τ x = τ x – τ s.

Appendix

Appendix A
Device-specific Initial Setting/List of Initialization Conditions

Condition setting for measuring instruments and initial set values for parameters are shown in the following table. Whether initialization is performed by the initialization commands (*RST, INI) is also indicated.

Items whose *RST field is filled in by a circle : initialized by the *RST command
Items whose INI field is filled in by a circle : initialized by the INI command

Device-specific initial setting/list of initialization conditions

Group	Device parameter	Initial value	*RST	INI
CHANNELS				
MENU	Measuring channel	CH1	○	○
	Coupled channel	ON	○	○
ACTIVE	Active channel	CH1	○	○
MEAS	Measuring (analytical) port	TA/R	○	○
FORMAT	Measurement format	LOGMAG	○	○
	Impedance marker form	Z/θ	○	○
	Admittance marker form	Y/θ	○	○
	Phase offset	0 deg	○	○
MEASUREMENT				
FREQUENCY/SPAN	Center frequency	150 MHz	○	○
	Frequency span	100 MHz	○	○
	Start frequency	10 MHz	○	○
	Stop frequency	300 MHz	○	○
	Log start frequency	10 MHz	○	○
	Log stop frequency	300 MHz	○	○
	Frequency setting mode	START/STOP	○	○
	Linear/log mode	LINEAR	○	○
AVG	Averaging count	1	○	○
	Smoothing	0 %	○	○
	Delay aperture	0.4 %	○	○
	Resolution bandwidth	20 kHz (AUTO)	○	○
	Averaging type	SUM	○	○
SWEEP	Sweep mode	REPEAT	○	○
	Sweep time	75 ms (AUTO)	○	○
	Marker sweep	OFF	○	○
	Measuring point	501	○	○
	Breakpoint	1001	○	○
	External trigger	OFF (INT)	○	○
	External trigger mode	NORMAL	○	○
	External trigger edge	RISE	○	○
OUT/INPUT	Output power	–6 dBm	○	○
	Source power	–6 dBm	○	○
	Output ATT (option)	0 dB	○	○
	Output offset	0 dB	○	○

Device-specific initial setting/list of initialization conditions

Group	Device parameter	Initial value	*RST	INI
OUT/INPUT	Power sweep	OFF	○	○
	Start power	0 dBm	○	○
	Stop power	0 dBm	○	○
	Step	0 dB	○	○
	Output port	OUTPUT-B	○	○
	Input range-R	0 dBm	○	○
	Input range-TA	0 dBm	○	○
	Input range-TB (option)	0 dBm	○	○
	Input impedance-R	50/75 Ohm	○	○
	Input impedance-TA	50/75 Ohm	○	○
	Input impedance-TB (option)	50/75 Ohm	○	○
	Standard impedance	50 Ohm	○	○
CALIBRATION				
CAL	Calibration type	RESPONSE	○	○
	Through line offset length	0.0 mm	○	○
	Open device parameter	All 0.0	○	○
	Open device offset length	0.0 mm	○	○
	Short device offset length	0.0 mm	○	○
	Calibration data taking in	DEFAULT	○	○
	Normalize (X-S)	OFF	○	○
	S memory data	Indefinite	○	○
DISPLAY				
SACLE	Scale	10 dB/	○	○
	Offset	–50.0 dB	○	○
	Offset line	5 (CENTER)	○	○
ACTIVE TRACE	Active trace	TRACE-A	○	○
TRACE	Split display	OFF	○	○
	Waveform storage	OFF	○	○
	Overwrite	OFF	○	○
	Grid type	ALL	○	○
	Display item	SETUP-A:ON SETUP-B:ON MEAS PRMS:ON FREQ:OFF MENU:OFF SWEEP MKR:ON CHART-A:OFF CHART-B:OFF TRACE-A:OFF TRACE-B:OFF MARKER-A:ON MARKER-B:ON TOP LINE:ON	○	
	Subtrace ON/OFF	OFF	○	○
	Subtrace type	MT=ST	○	○

Device-specific initial setting/list of initialization conditions

Group	Device parameter	Initial value	*RST	INI
MARKER				
MKR	Active marker number	0	○	○
	Number of ON markers	1	○	○
	Marker position	250 point	○	○
	Marker mode	NORMAL	○	○
	Reference marker number	0	○	○
	Marker list	OFF	○	○
FCTN	Marker function	All OFF	○	○
UTILITY				
FILTER	Filter analysis function	OFF	○	○
	Filter nominal center frequency	150 MHz	○	○
	Insertion loss calculation standard	FILTER CF	○	○
	Bandwidth calculation standard	FILTER CF	○	○
	X1 dB drop band X1 dB	3.0 dB	○	○
	X2 dB drop band X2 dB	60.0 dB	○	○
	Ripple search start frequency	10 kHz	○	○
	Ripple search end frequency	300 MHz	○	○
	Ripple resolution	0.02 dB	○	○
	Frequency result display place number	6	○	○
	Analysis result	All 0	○	○
RESONATOR	Resonator analysis function	OFF	○	○
	Analysis types	RESON1 MIN/MAX IMPD	○	○
	Analysis band start frequency	10 kHz	○	○
	Analysis band end frequency	300 MHz	○	○
	Frequency result display place number	6	○	○
LIMIT	Analysis result	All 0	○	○
	Limit line test function	OFF	○	○
	Limit line type	SEGMENTED	○	○
	Single line standard value (Y)	0 dB (both upper and lower limits)	○	○
	Segment line standard value (X)	10 Hz, 300 MHz	○	○
	Segment line standard value (Y)	0 dB, 0 dB	○	○
	Test results	NO TEST	○	○
	BEEP ON/OFF	OFF	○	○

Device-specific initial setting/list of initialization conditions

Group	Device parameter	Initial value	*RST	INI
COPY&MEMORY				
COPY CONTROL	GPIB address	1		
	GPIB control function	DEVICE		
	ENABLE REGSTER ALL	OFF		
	Terminator	LF&EOI		
	Time-out	20 sec		
	Active port	GPIB		
	RS-232C control function	CONTROLLER		
	BAUD RATE	9600 bps		
	DATA BITS	8 bit		
	STOP BIT	1		
	PARITY	OFF		
	Device for copying	VIDEO OUT		
SAVE/RECALL	SAVE ITEM	Only PARAM for both channel 1 and channel 2 : ON	○	○
	Drive selection	FD	○	
	Title display	OFF	○	○
	Title registering character	null	○	○
SYSTEM				
USER PRESET	Marker setting mode	POINT	○	
	Impedance measurement type	TRANSFER	○	
COLOR	Display color (1st screen)	Red	○	
	Display color (2nd screen)	White	○	
	Display color (3rd screen)	White	○	
	Display color (4th screen)	Blue	○	
	Display color (5th screen)	Pale blue	○	
	Display color (6th screen)	Purple	○	
	Display color (7th screen)	White	○	
	Display color (8th screen)	Green	○	
	Display color (9th screen)	Yellow	○	
	Display color (10th screen)	Dark green	○	
	Display color (11th screen)	Dark yellow	○	
	Display color (12th screen)	Red	○	
	Display color (13th screen)	Dark white	○	
	Display color (14th screen)	Dark white	○	
	Display color (15th screen)	White	○	
	Display color (16th screen)	White	○	
	Display color (background screen)	Dark black	○	
PTA	PTA ON/OFF	OFF	○	
BASK LIGHT	BACK LIGHT ON/OFF	ON	○	○

Appendix B ASCII Code List

B7 B6 B5 / BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 SP 20 32	60 0 30 48	100 @ 40 64	120 P 50 80	140 、 60 96	160 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 ! 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a 61 97	161 q 71 113
0 0 1 0	2 NUL 2 2	22 DC2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b 62 98	162 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 # 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c 63 99	163 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 S 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 d 64 100	164 t 74 116
0 1 0 1	5 PPC ENO 5 5	25 PPU NAK 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e 65 101	165 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 & 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f 66 102	166 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g 67 103	167 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 (28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h 68 104	170 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51) 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i 69 105	171 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 j 6A 106	172 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 ÷ 2B 43	73 ; 3B 59	113 K 4B 75	133 [5B 91	153 k 6B 107	173 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \ 5C 92	154 l 6C 108	174 ¦ 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 — 2D 45	75 = 3D 61	115 M 4D 77	135] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 ∧ 5E 94	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47	77 ? 3F 63	117 O 4F 79	137 UNT _ 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	Address command	Universal command	Listen address		Talk address		Secondary address or command	

KEY

octal	25 PPU	GPIB code
	NAK	ASCII character
hex	15 21	decimal

*USA Standard Code for Information Interchange

Table B-1 GPIB interface messages (extended version)

Bits				b7 → b6 → b5 →	0 0 0	① MSG	0 0 1	MSG	0 1 0	MSG	0 1 1	MSG	1 0 0	MSG	1 0 1	MSG	1 1 0	MSG	1 1 1	MSG
② ↓	b3 ↓	b2 ↓	b1 ↓	COLUMN → ROW	0		1		2		3		4		5		6		7	
0	0	0	0	0	NUL		DLE		SP	↑	0	↑	@	↑	P	↑	`	↑	p	↑
0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
0	1	0	1	5	ENQ	PPC	NAK	PPU	%		5		E		U		e		u	
0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	7	BEL		ETB		’		7		G		W		g		w	
1	0	0	0	8	BS	GET	CAN	SPE	(	Listener address (MLA) assigned to equipment	8	Listener address (MLA) assigned to equipment	H	Listener address (MTA) assigned to equipment	X	Listener address (MTA) assigned to equipment	h	Meaning defined by PCG	x	Meaning defined by PCG
1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1	0	1	0	A	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	B	VT		ESC		+		;		K		[		k		{	
1	1	0	0	C	FF		FS		,		<		L		\		l		¦	
1	1	0	1	D	CR		GS		–		=		M		]		m		}	
1	1	1	0	E	SO		RS		.		>	↓	N		^	↓	n		~	↓
1	1	1	1	F	SI		US		/	↓	?	UNL	O	↓	—	UNT	o	↓	DEL	

Address Command Group (ACG)

Universal Command Group (UCG)

Listen Address Group (LAG)

Talk Address Group (TAG)

Primary Command Group (PCG)

Secondary Command Group (SCG)

Notes:
① MSG=INTERFACE MESSAGE (Sent by ATN of True, Low level)
② b1=DI01...b7=DI07 (b1 through b7 correspond to DI01 to DI07 sequence.)GTL

SDC	Go to Local
PPC	Select Device Clear
GET	Parallel Poll Configure
TCT	Group Execute Trigger
LLO	Take Control
DCL	Local Lockout
PPU	Device Clear
SPE	Parallel Poll Unconfigure
SPD	Serial Poll Enable
UNL	Serial Poll Disable
UNT	Unlisten
(ACG)	Untalk
(UCG)	Addressed Command Group
(LAG)	Universal Command Group
(TAG)	Listen Address Group
(PCG)	Talk Address Group
(SCG)	Primary Command Group
	Secondary Command Group

Table B-2 Interface message group

DI08	DI07	DI06	DI05	DI04	DI03	DI02	DI01	Interface message group (G)
×	0	0	0	b4	b3	b2	b1	Addressed command G
×	0	0	1	b4	b3	b2	b1	Universal command G
×	0	1	b5	b4	b3	b2	b1	Listen address G
×	0	1	1	1	1	1	1	Unlisten (UNL)
×	1	0	b5	b4	b3	b2	b1	Talker Address G
×	1	0	1	1	1	1	1	Untalk (UNT)
×	1	1	b5	b4	b3	b2	b1	Secondary command G

Table B-3 Address allocation table

Address character		Address swich setting					Primary address	Factory address set device
Talk	Listen	5	4	3	2	1		
b_7 b_6	b_7 b_6	b_5	b_4	b_3	b_2	b_1		
1 0	0 1	↓	↓	↓	↓	↓	Decimal	
@	SP	0	0	0	0	0	0	
A	!	0	1	0	0	1	1	
B	"	0	0	0	1	0	2	
C	#	0	0	0	1	1	3	
D	$	0	0	1	0	0	4	
E	%	0	0	1	0	1	5	
F	&	0	0	1	1	0	6	
G	'	0	0	1	1	1	7	
H	(	0	1	0	0	0	8	
I	)	0	1	0	0	1	9	
J	*	0	1	0	1	0	10	
K	+	0	1	0	1	1	11	
L	,	0	1	1	0	0	12	
M	-	0	1	1	0	1	13	Printer
N	.	0	1	1	1	0	14	Plotter
O	/	0	1	1	1	1	15	
P	0	1	0	0	0	0	16	
Q	1	1	0	0	0	1	17	
R	2	1	0	0	1	0	18	
S	3	1	0	0	1	1	19	
T	4	1	0	1	0	0	20	
U	5	1	0	1	0	1	21	
V	6	1	0	1	1	0	22	
W	7	1	0	1	1	1	23	
X	8	1	1	0	0	0	24	
Y	9	1	1	0	0	1	25	
Z	:	1	1	0	1	0	26	
[	;	1	1	0	1	1	27	
\	<	1	1	1	0	0	28	
]	=	1	1	1	0	1	29	
^	>	1	1	1	1	0	30	
?	–	1	1	1	1	1	31	UNL,UNT

Appendix C
Comparison Table of GPIB Commands for Controllers

Function	Controller				
	PACKET V	PC9800	IBM-PC (NI-488.2)	IBM-PC (NI-488)	HP9000 series
Outputs data to a device	WRITE @ device number: data	PRINT @ listener address; data	CALL Send( )	CALL IBWRT( )	OUTPUT device selector; data
Output binary data to a device	BIN WRITE @ device number: data	WBYTE command; data	CALL SEND Cmds( )		
Assigns data entered from a device to a variable	READ @ device number: variable	INPUT @ talker address, listener address; variable LINE INPUT @ talker address, listener address; variable	CALL Receive( )	CALL IBRD( )	ENTER device selector; variable
Assigns binary data entered from a device to a variable	BIN READ @ device number: variable	RBYTE command; variable			
Initializes an interface	IFC @ select code	ISET IFC	CALL Send IFC( )	CALL IBSIC( )	ABORT select code
Turns REN line on	REN @ select code	ISET REN	CALL Enable Remote( )	CALL IBSRE( )	REMOTE device selector (select code)
Turns REN line off	LCL @ select code (sets all devices local) LCL @ device number (sets only specified devices to listeners, and sends out GTL command)	IRESET REN, WBYTE & H3F, listener address, &H01	CALL Enable Local( )	CALL IBSRE( ) CALL IBLOC( )	LOCAL device selector (select code) LOCAL device selector (select code + primary address)
Outputs interface message(s) and data	COMMAND @ select code: Character string for message [;data]			CALL IBCMD( ) CALL IBCMDA( ) (asynchronous)	SEND select code; message string
Triggers a specified device	TRG @ device number	WBYTE & H3F, listener address, secondary address, &H08;	CALL Trigger( )	CALL IBTRG( )	TRIGGER device selector

Function	Controller				
	PACKET V	PC9800	IBM-PC (NI-488.2)	IBM-PC (NI-488)	HP9000 series
Initializes devices	CDL @ select code (all devices having a specified select code) DCL @ device number (specified devices only)	WBYTE &H3F, &8H14; WBYTE &H3F, listener address, secondary address, &H04	CALL DevClear( )	CALL IBCLR( )	CLEAR device selector (select code) CLEAR device selector (select code + primary address)
Prevents a device from being switche d over from remote to local	LLO @ select code	WBYTE &H3F, &H11;	CALL SendLLO( ) CALL SetRWLS( )	LOCAL LOCKOUT	
Transfers control to a specified device	RCT @ device number	WBYTE talker address, &H09;	CALL Pass Control( )	CALL IBPCT( )	PASS CONTROL
Sends out a service request	SRQ @ select code	ISET SRQ		CALL IBRSV( )	REQUEST select code
Performs serial polling	STATUS @ device number	POLL	CALL Read Status Byte( ) CALL AllSpoll( )	CALL IBRSP( )	SPOLL (device selector) (function)
Sets a terminator code	TERM IS	CMD DELIM		CALL IBEOS( ) CALL IBEOT( )	
Sets a limit value for checking a time-out		CMD TIMEOUT		CALL IBTOM( )	
Wait to SRQ			CALL WaitSRQ( )	CALL IBWAIT( )	

MS4630B
Network Analyzer
Remote Operations (PTA Control)
Operation Manual

Contents

Section 1 Outline

The Personal Test Automation (PTA) is a measuring device equipped with a programming language interpreter function. The PTA allows you to program the controls and calculations directly connected to a measuring system using a simple language named the Personal Test Language (PTL). The PTL consists of basic commands similar to the BASIC language, as well as GPIB control commands, file operation commands, screen control commands and function control commands that control most of the functions related to the measurements.

You can create, edit and execute a PTA program on the screen of this device using a full-scale keyboard. Alternatively, you can create and edit a PTA program on an external personal computer using a general-purpose editing program, and then register via the GPIB and execute it on this device.

The PTA is provided with such external interfaces as the GPIB, RS-232C, parallel (Centronics), and PTA parallel I/O port. The GPIB and RS-232C, used to connect this device to an external computer, allow the PTA and the external computer to communicate with each other. The PTA parallel I/O port, used to connect this device to an automatic checker or trimmer of electronic parts, allows the PTA to control these machines.

1.1 PTA Specifications

The following shows the PTA specifications.

■ Screen

• Number of displayable characters:	68 characters x 30 lines
• Displayable character types:	Alphabet in upper and lower cases, numbers, special characters, and cursor
• Fonts:	8×16 dots (small font)
• Graphic display:	Straight lines, squares, circles, and arcs
• Screen:	640×480 dots $\times$ 16 screens

■ Input and execution control

• Input:	Full-scale keyboard, front panel, and external computer (GPIB)
• Execution control:	Full-scale keyboard, front panel, and external computer (GPIB)

■ Memory

• Program memory:	148 kilobytes

■ Language, version PTL-V1.6

• Commands:	Editing commands Program execution control commands File commands
• Statements:	Basic statements GPIB statements/PTA parallel I/O port statements Dual port statements
• Subroutines	Screen subroutines File operation subroutines Video plotter subroutines GPIB subroutines Interface subroutines Panel subroutines Waveform memory copy subroutines Buzzer subroutines
• Functions:	Arithmetic functions Logical functions Statistical functions Character string functions System functions

• Variables	User-defined variables (numeric variables and character string variables) up to 256 System variables
• Data types	Real numbers: Significant digits: 15, Exponents: 10^{308} to 10^{-307} Integers: –32768 to +32767 Characters: Max. 255 Bits: Max. 8

■ Interfaces

• GPIB	
• RS-232C (Option 02)	
• Parallel (Centronics) (Option 02)	
• PTA parallel I/O port	Output port A: 8 bits Output port B: 8 bits Input/output port C: 4 bits Input/output port D: 4 bits Control port: 3 bits

1.2 PTL Provided in the PTA

The following shows the Personal Test Language (PTL) provided in the PTA.

Item	Format
■ Editing commands	
• Entering a program	Line number Statement
• Automatically updating the line numbers	AUTO [Top line number] [, Increment]
• Copying	PCOPY New top line number, [Increment], Source start line number, Source end line number
• Deleting	DELETE [Delete start line number] [, Delete end line number]] or [Delete line number] [RETURN]
• Renumbering	RENUN [New top line number[, Increment[, Old start line number[Old end line number]]]
• List output (screen)	LIST [Start line number][, [End line number]]
• List output (printer)	LISTG Address[, Start line number][, [End line number]]
• Program size	PMEMO
• Page scrolling / Reverse page scrolling	[PAGE SCROLL] key / [SHIFT] + [PAGE SCROLL] key
• Rolling up / Rolling down	[CTRL] + [J] key / [CTRL] + [K] key
■ Execution command	
• Starting to execute the program	[RUN] menu key, RUN[Start line number][Suspend line number]
• Suspending executing the program	[STOP] menu key
• Continuing executing the program	[CONT] menu key, CONT[, Suspend line number]
• Canceling executing the program	[RESET] menu key
• Executing immediately	Statement[RETURN]
■ File commands	
• Saving a file	SAVE Program name[, Start line number[, End line number]]
• Overlay	OVERLAY
• Loading a file	LOAD Program name
• Displaying the file list	[PLIST]menu key
• Deleting a file	PDEL Program name
• Startup registration	SATRTP Program name
• Startup cancellation	CANCEL
• Saving a program in other measuring devices	SAVEG M
• Loading a program from other measuring devices	LOADG M

Item	Format
■ Statements	
• Remarks	REM ["Comment"] or 'Comment
• Array declaration	DIM Array variable
• Assignment	[LET] Variable=Equation(Function, Variable or Constant)
• Branch	GOTO Line number or GOTO *Label
• Branch to a subroutine	GOSUB Line number or GOSUB *Label
• Returning from a subroutine	RETURN
• Decision	IF Condition Statement
• Iteration start	FOR Numeric variable=Initial value TO End value [STEP Step value]
• Iteration end	NEXT Numeric variable
• Input	INPUT [“Displayed string”,] Variable [, Variable...]
• Display	PRINT Variable [: Data format] [, Variable [:Data format]...] [;]
• Reverse display	PRINTR Variable [: Data format] [, Variable [:Data format]...] [;]
• GPIB input	READ Address, Input variable
• GPIB input (1 byte)	BREAD Address, Input variable
• GPIB input (2 bytes)	WREAD Address, Input variable
• GPIB output	WRITE Address, Variable [:Data format] [;]
• GPIB output (1 byte)	BWRITE Address, Variable [:Data format]
• GPIB input (2 bytes)	WWRITE Address, Variable [:Data format]
• Defining a measurement parameter	PUT Character string variable (or Character constant)
• Reading the measurement parameter (1)	GET Character string variable (or Character constant), Input variable
• Reading the measurement parameter (2)	COM Character string variable (or Character constant)>Input variable
• Waiting	WAIT Time (in seconds, min. 10 ms)
• Calling a subroutine	CALL Subroutine name
• Moving the cursor (to home position)	HOME
• Moving the cursor	LOCATE (X, Y)
• Clearing the screen	ERASE
• Terminating the program	STOP
• Syntax error statement	Line number SOS Syntax error statement
• Error branch	ERROR (Error number, line number or label)
• Error main	ERRMAIN
• Returning the main routine	RETMAIN

Item	Format
• Initializing a variable	CLEAR
• Data statement	DATA Constant[, Constant..., Constant]
• Specifying the input data statement	RESTORE [Line number or *Label]
• Entering a data statement	RDATA Variable[, Variable...]
• Reading/executing a program	CHAIN “File name”
• Registering an error interrupt routine	ON ERROR Line number or Label
• Canceling the registration of an error interrupt routine	OFF ERROR
• Returning from an error interrupt	RETERR RETRY RESUME Line number or *Label GIVEUP
• Registering an event interrupt routine	ON EVENT I/O number, Line number or *Label
• Enabling an event interrupt	ENABLE EVENT I/O number, Factor 3, Factor 2, Factor 1, Factor 0
• Disabling an event interrupt	DISABLE EVENT I/O number[, Factor 3, Factor 2, Factor 1, Factor 0]
• Returning from an event interrupt	RETINE
• Interrupt branch (I/O port)	ON IO GOTO Line number or *Label
• Interrupt subroutine branch (I/O port)	ON IO GOSUB Line number or *Label
• Enabling an interrupt (I/O port)	IOEN
• Disabling an interrupt (I/O port)	IODI
• Interrupt mask (I/O port)	IOMA
• Specifying the character size	DCHSIZE Character size number
• Defining the pseudo-random number sequence	RNDMIZE
• Changing the output timing of the parallel I/O port write strobe.	OLDPORT
■ Dual port memory statements	
• Writing data	WDPM Memory number, Variable [:Data format]
• Reading data	RDPM Memory number, Input variable
■ Screen subroutines	
• Deleting a display item	CALL CER(M)
• Returning a display item	CALL CRN(M)
• Clearing the screen	CALL CFL(M)
• Displaying characters	CALL DCH(X, Y, text, M[, N])
• Drawing a line	CALL DLN(X0, Y0, X1, Y1, M[, N])
• Drawing a square	CALL DRC(X0, Y0, X1, Y1, M[, N])

Item	Format
• Drawing a circle	CALL DCR(X, Y, R, M[, N])
• Drawing an arc	CALL DAR(X0, Y0, R0, W1, W2, M[, N])
• Drawing a OX	CALL DJG(X, Y, R, F, M1, M2)
• Registering a soft key label	CALL DEF(M, text)
■ File operation subroutines	
• Opening a file (reading)	CALL OPNI Character string variable (or Character constant)
• Opening a file (writing)	CALL OPNO Character string variable (or Character constant)
• Deleting a file	CALL FDEL Character string variable (or Character constant)
• Loading data	CALL DALD Variable
• Saving data	CALL DASV Variable
• Closing a file	CALL CLS
■ Video plotter subroutine	
• Hard copying the screen	CALL VPT
■ Panel subroutines	
• Locking the operation of front panel	CALL PNLL(0)
• Unlocking the operation of front panel	CALL PNLU(0)
■ Waveform memory copy subroutines	
• Memory copy	CALL COPY(M0, M1)
■ GPIB subroutines	
• Clearing the interface (Switching to the system controller port)	CALL IFC
• Service request	CALL RSV(M)
• Switching to the device port	CALL DEV
■ Interface subroutine	
• Interface control	CALL GPIB (Port number, Control item number)
■ Buzzer subroutine	
• Buzzer calling	CALL BZR
■ Functions	
• Arithmetic functions	SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, SQR, ABS, SGN, INT, ROUND, DIV, FIX
• Logical functions	NOT, AND, OR, EOR
• String functions	CHR, VAL, HVAL, BVAL, ASC, CHR$, CVI, CVD, MKI$, MKD$, STR$, HEX$, OCT$, BIN$, INSTR, LEFT$, MID$, RIGHT$, STRING$, LEN, SLEN, SGET$

Item	Format
• Statistical functions	max, min, sum, mean, var, sta
• Dedicated functions	ERRREAD, STATUS, DTREAD$, RND
■ System variables	
	EX0, EX1, EX2, EX3, EX4, EX5, EX6, DT0, DT1, DT2, DT3, DT4, XMA, XMB, SMA, SMB, IMA, IMB, IDA, IDB, FQM, LVM, IOA, IOB, IOC, IOD, EIO, TRC, CAL, CXS, MKR, RMK, CMK, MKD, ZRA, ZRB, RFA, RFB, MVA, MVB, MKF, STR, SAU, SCA, SCB, OFA, OFB, DF1, DF2, CNF, SPF, STF, SOF, FRQ, SWT, LGF, IRG, OPL, SOL, SEL, LSW, AU1, AU2, RBW, IMP, OVL, SW1, SW2, SW3, SWP, AVG, MEP, BKP, TTL, OVP, INI, SV1, SV2, SV3, SV4, SV5, DLM, MA4, TRM, GTM, PHO, FI1, FO1, FI2, FO2, FI3, FO3, FI4, FO4, FI5, FO5
■ System functions	
• Maximum value	MAX(M, P0, P1, N)
• Minimum value	MIN(M, P0, P1, N)
• Specified measurement value frequency 1	BNDL(M, P0, L, N)
• Specified measurement value frequency 2	BNDH(M, P0, L, N)
• Specified measurement value frequency 3	MEAS(M, P0, L, N)
• Ripple 1	RPL1(P0, P1, N[, R])
• Ripple 2	RPL2(P0, P1, N[, R])
• Ripple 3	RPL3(P0, P1, N[, R])
• Ripple 4	RPL4(P0, P1, N[, R])
• Polling 1	POLL(M, P0, L, N[, R])
• Polling 2	POLH(M, P0, L, N[, R])
• Peak value	PLRH(M, P0, N[, R])
• Trough value	PLRL(M, P0, N[, R])
• Filter 1	FILTER1(N)
• Filter 2	FILTER2(N)
• Filter 3	FILTER3(N)
• Filter 4	FILTER4(N)
• Filter 5	FILTER5(N)
• Group delay	GPDLY(P0, P1, CH[, E])
• Reading the status byte	GST(M)

PTA Parallel I/O Port Connectors (RC30-36R: Manufactured by Hirose Electric Co., Ltd.)

Pin No.	Name	Specification	System variable name
1	GND		
2	INPUT 1	TTL level, negative logic	EX0
3	OUTPUT 1	TTL level, negative logic	EX0
4	OUTPUT 2	TTL level, negative logic	EX0
5	Output port A0	TTL level, negative logic	IOA
6	Output port A1	TTL level, negative logic	IOA
7	Output port A2	TTL level, negative logic	IOA
8	Output port A3	TTL level, negative logic	IOA
9	Output port A4	TTL level, negative logic	IOA
10	Output port A5	TTL level, negative logic	IOA
11	Output port A6	TTL level, negative logic	IOA
12	Output port A7	TTL level, negative logic	IOA
13	Output port B0	TTL level, negative logic	IOB
14	Output port B1	TTL level, negative logic	IOB
15	Output port B2	TTL level, negative logic	IOB
16	Output port B3	TTL level, negative logic	IOB
17	Output port B4	TTL level, negative logic	IOB
18	Output port B5	TTL level, negative logic	IOB
19	Output port B6	TTL level, negative logic	IOB
20	Output port B7	TFL level, negative logic	IOB
21	Input/Output port C0	TTL level, negative logic	IOC
22	Input/Output port C1	TTL level, negative logic	IOC
23	Input/Output port C2	TTL level, negative logic	IOC
24	Input/Output port C3	TTL level, negative logic	IOC
25	Input/Output port D0	TTL level, negative logic	IOD
26	Input/Output port D1	TTL level, negative logic	IOD
27	Input/Output port D2	TTL level, negative logic	IOD
28	Input/Output port D3	TTL level, negative logic	IOD
29	Port C status	TTL level, 0/1:Input/Output	EIO
30	Port D status	TTL level, 0/1:Input/Output	EIO
31	Write strobe signal (*1)	TTL level, negative logic	
32	Interrupt signal (*2)	TTL level, negative logic	
33	NC		
34	+5V output	Max. 100 mA	
35	NC		
36	NC		

Note:

1. This signal is output when the IOC=... or IOD=... statements are executed.
2. The execution for the interrupt will be performed by the I/O interrupt statements ON IO GOTO ··· and ON IO GOSUB ···

CAUTION ⚠

For details of system variables, see CHAPTER 9, "CONTROLLING THE PTA PARALLEL PORT".

The name of the above connector is RC30-36R manufactured by Hirose Electric Co., Ltd. Have handy also the RC-30P manufactured by Hirose Electric Co., Ltd. as this is the connector to be engaged to RC30-36R.

1.3 External interfaces

This measuring device comes standard with the GPIB interface and the PTA parallel I/O port interface. The RS-232C/ Centronics (printer) interface can be optionally added (option 02). These interfaces are used in different ways depending on the settings of the connection ports.

1.3.1 GPIB Interface

- If "Control Function" in the GPIB setting condition of this device is set to "DEVICE":

  The GPIB interface will act as a device port, to which the host computer or other control unit for this device can be connected.

- If "Control Function" in the GPIB setting condition of this device is set to "CONTROLLER":

  The GPIB interface will act as a system controller port.
  Execute the COPY command from the PTA to print a screen copy on the printer.

  You can also control the external devices from the PTA.

1.3.2 RS-232C Interface

- If "Control Function" in the RS-232C setting condition of this device is set to "DEVICE":

  The host computer or other control unit for this device can be connected to the RS-232C interface.

- If "Control Function" in the RS-232C setting condition of this device is set to "CONTROLLER":

  Execute the COPY command from the PTA to print a screen copy on the printer.

  The serial data transfer can be performed between the PTA and the external devices.

1.3.3 Parallel (Centronics) Interface

Execute the COPY command from the PTA to print a screen copy on the printer.

1.3.4 PTA Parallel I/O Interface

Use this interface to control the devices not having a GPIB or RS-232C interface or the devices not provided with a special protocol or handshake for data transfers.
Using a PTA statement will facilitate controlling the external devices.

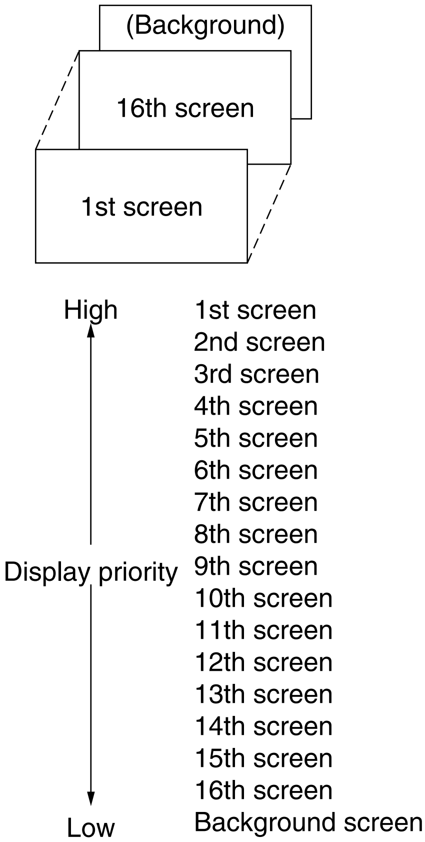
Notes:

- Only one of the GPIB and the RS-232C interfaces can be set as a device port.
- Both the GPIB and the RS-232C interfaces can be set as a controller port.
- To print a hard copy or control an external device through the GPIB, RS-232C, or parallel (Centronics) interface, you must select which of the interface ports the data should be output through. Make the selection in a window called up from the GPIB, RS-232C, or HARD COPY menu of this device, or using the GPIB command "PORTn". The selected interface port will be called an active port. An active port is enabled only if the interface is set as a controller port. The parallel (Centronics) interface is always used as a controller port.

1.4 Structure of Screens of the Measuring Device

Section 1.4 describes the screen specification of this device.

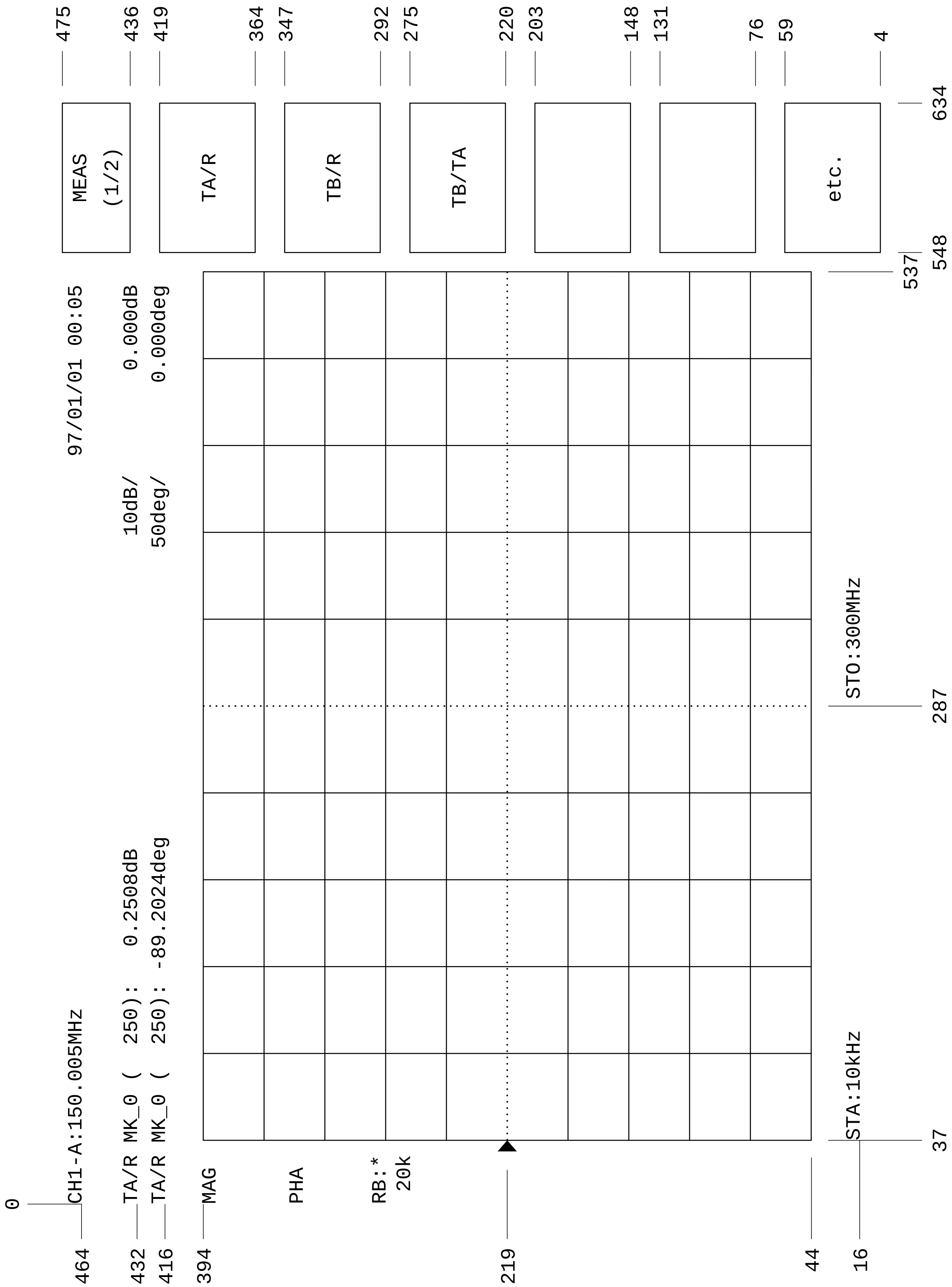
1.4.1   Physical Structure of Screens



Note:

For information on what is displayed on each screen, see the chapter on screen subroutines (CALL CFL).

1.4.2 Display Coordinates of Measurement Screen



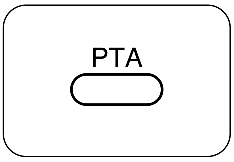
Section 2 Operating the PTA

2.1 Operation Overview

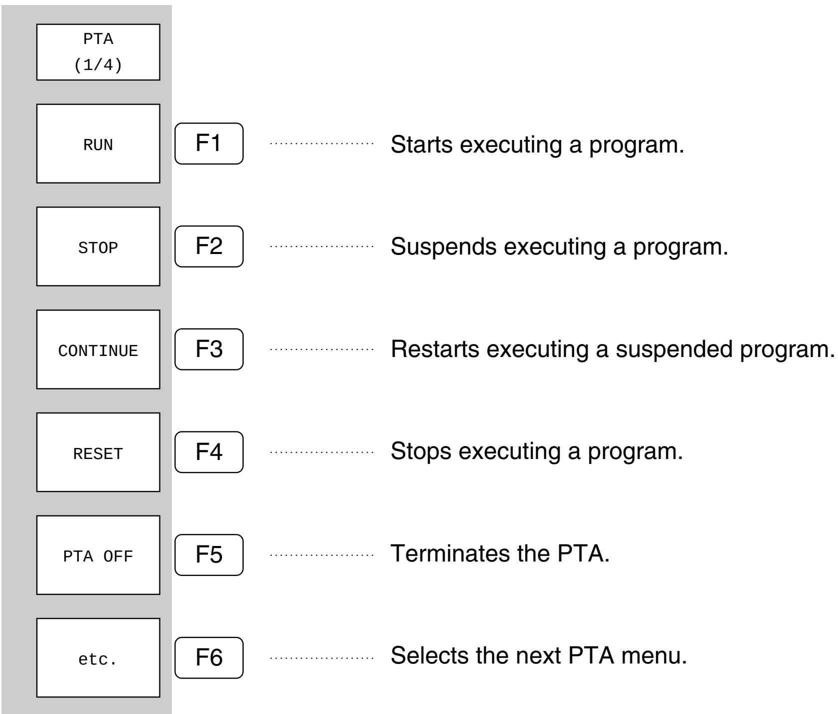
In the PTA, an external keyboard is used to enter, edit and execute a program. In addition, the front panel of this device or the GPIB can be used to input and execute a program.

2.2 Starting the PTA

Press the [PTA] key on the front panel of this device to turn on the PTA.



Then, the display screen will be cleared and the cursor will appear at the HOME position (upper left corner of the screen). Press the soft keys [F1] to [F6] to display the corresponding PTA menus.



Notes:

- Pressing [F5] PTF OFF will not delete the PTA program from the memory. By pressing the [PTA] key again to turn on the PTA, you can use the program immediately.
- Loading a new program after turning on the PTA will delete the previous program.
- To load a program without deleting the previous program but executing overwrite, execute the OVERLAY command and then load the program.
- <u>Turning off the power of this device will delete the PTA program in the memory.</u>

2.3 Panel Operation

2.3.1 Soft Keys

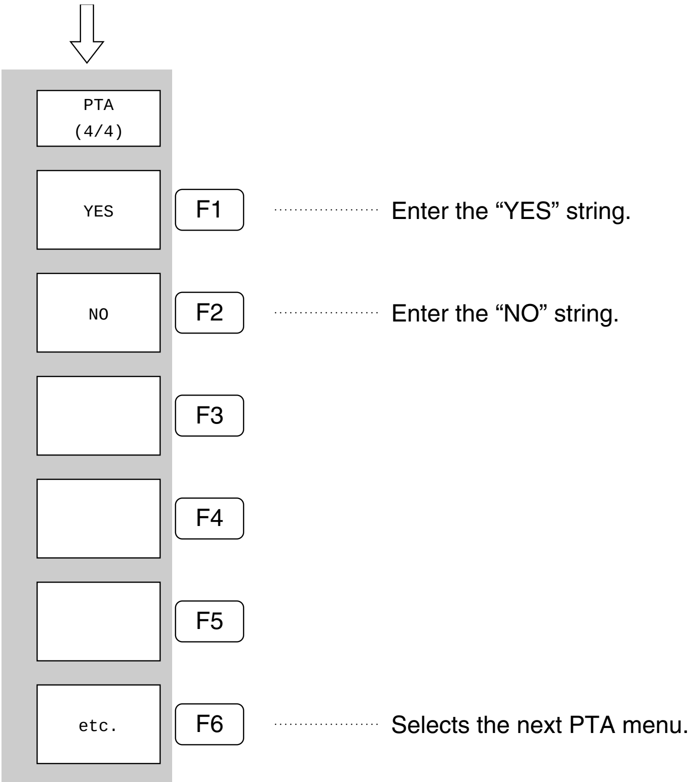
After the PTA is turned on, pressing the [F6] "etc." soft key to display soft key menus in turns.

PTA (2/4)		
PROG LIST	F1	Displays the list of PTA file names stored in the storage device (internal memory, FD or PMC).
CURSOR UP	F2	Moves up the cursor.
CURSOR DWN	F3	Moves down the cursor.
LOAD	F4	Loads the program with the file name indicated in the cursor line.
RUN	F5	Starts executing a program.
etc.	F6	Selects the next PTA menu.

⇩

PTA (3/4)		
F1	F1 †	A system switch that can be read in the PTA program.
F2	F2 †	A system switch that can be read in the PTA program.
F3	F3 †	A system switch that can be read in the PTA program.
F4	F4 †	A system switch that can be read in the PTA program.
F5	F5 †	A system switch that can be read in the PTA program.
etc.	F6 †	Selects the next PTA menu.

† Use the DEF subroutine to define the displayed characters.



2.3.2 Entering, Executing and Stopping the Program

The program entered and edited using the external keyboard can be saved into the storage device (internal memory, FD or PMC). The following section shows the procedures for loading and executing, through the front panel operations, the program saved in the storage device.

(1) Display, in the list format, the names of the programs in the storage device by pressing the [F1] PROG LIST key. If you have just finished pressing the [PTA] key, press [F6] etc once instead to display PROG LIST for [F1] of the soft key, before starting the operations.

(2) Press the [F2] CURSOR UP or [F3] CURSOR DWN key to move the cursor to the name of program to be loaded.

(3) Press the [F4] LOAD key to load the program.

(4) Press the [F5] RUN key to start executing the program.

.
.
.
.

(5) Press the [F5] RESET key to stop executing the program.

Select the storage device (internal memory, FD or PMC) in the "DRIVE" menu called up by pressing the SAVE/RECALL key of the main frame function.

2.3.3 Data Input Keys

Use the soft and numeric keys as data input keys.

(1) [F1]F1, [F2]F2, [F3]F3, [F4]F4, and [F5]F5 keys

These keys can be referenced on the program and correspond to the system variables EX1, EX2, EX3, EX4 and EX5, respectively. Pressing a key will change the variable data to 0 and 1 each time. In the initial or reset state, all the variables will be 0. Use the DEF subroutine to define the label names for the menu (F1 to F6).

Note:

For EX1, EX2, EX3, EX4 and EX5, see “System Variables” in CHAPTER 5.

(2) [F1]YES and [F2]NO keys

These keys are the input keys for the INPUT statement. Press these keys to enter the YES and NO strings immediately.

(3) Numeric keys

The numeric keys for ENTRY are the input keys for the INPUT statement.
Use the [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], and [·] keys to enter a numeric. Use the [+/-] or [MHz] key to enter a minus symbol (–) and the [KHz] key to enter a comma (,). To correct an entered numeric, delete the characters using the [BS] key and re-input a correct numeric. To end the input, press the [ENTER] key.

2.3.4 Other Panel Key Operations

When the PTA is turned on, all the panel keys are locked except for the [PTA], [Local], numeric/[MHz]/[kHz]/[BS]/[ENTER], [∧], [∨], [<], [>] keys and the [F1] to [F6] soft keys. To use the front panel keys to define the parameters of this device while the PTA is on (including when a program is executed), execute CALL PNLU (0) to unlock all the panel keys. (See the section on PNLU subroutines).

2.4 Terminating the PTA

To terminate the PTA, first press the [RESET] soft key to stop the execution of a program, and then press the [PTA OFF] key. The screen displayed by the display subroutine will be cleared and the normal measurement screen will be displayed.

Note:

For the display subroutine, see “System Subroutines” in CHAPTER 5.

Precaution during PTA program execution

The device uses the same micro processor to execute the PTA program as well as execute measurements and control the sweep. Therefore, it is disabled to execute the PTA program as well as execute measurements and control the sweep at the same time. Moreover, the priority of the job execution is set as follows; execution of sweep < execution of the PTA program < execution of setup such as the measuring conditions. When a job of higher priority is being executed, execution of other jobs of lower priority must queue.

For example, when the PTA program is being executed, execution of regular sweeping must queue. However, execution of the PTA program is temporarily paused when PTA has sent a command to execute sweep and is now waiting for the response for execution completion, or when the waiting statement (WAIT) is being executed. In this case, sweeping is executed. When PTA is executing statements other than the one mentioned above, execution of sweeping must generally queue.

2.5 External Keyboard

Use an external keyboard to enter, edit and execute a program.

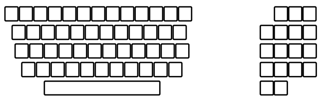
2.5.1 Connecting an External Keyboard

Connect an external keyboard to 8-pin round DIN connector at [KEYBOARD] of the front panel of this device.

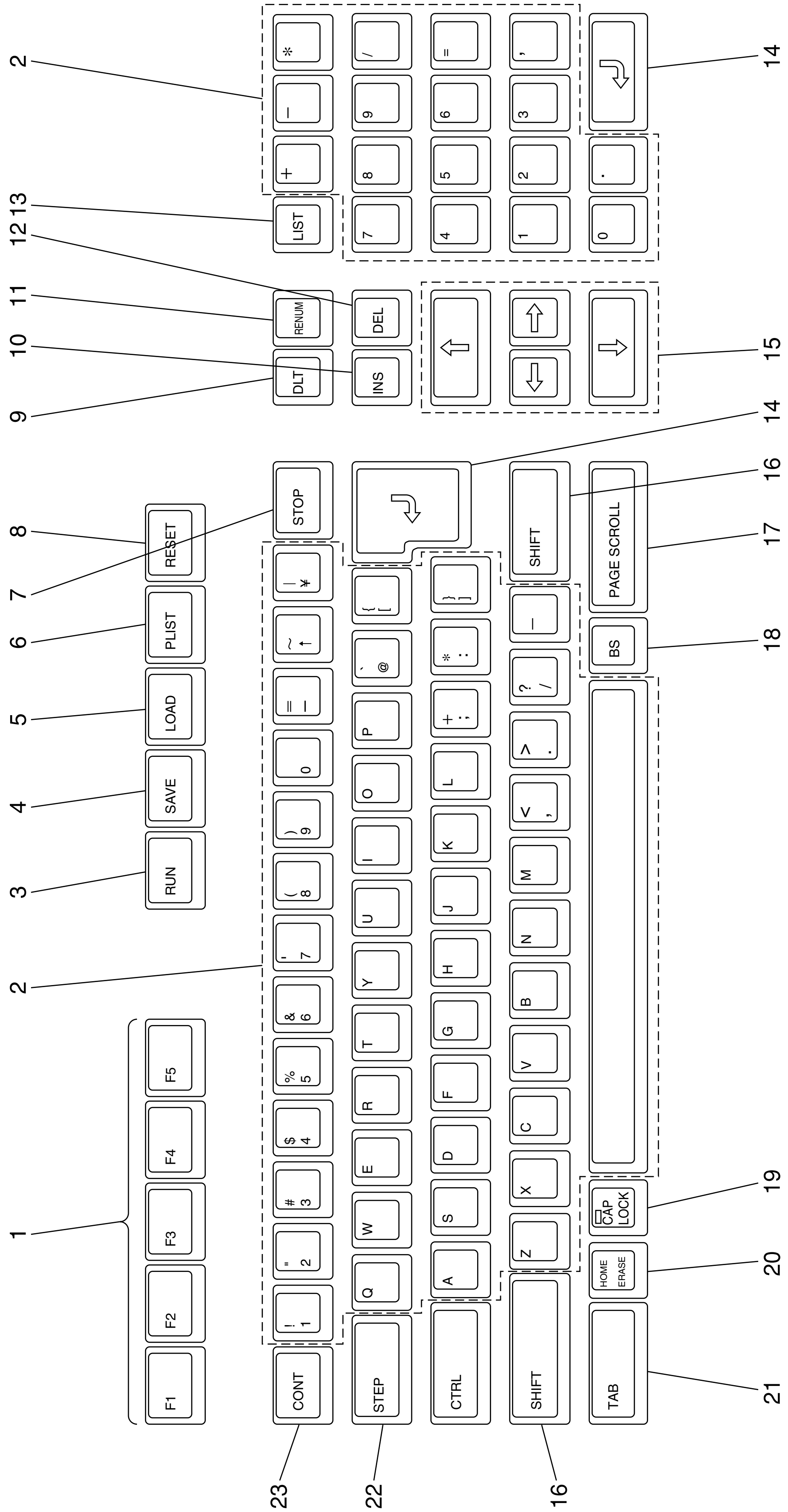
Before connecting or disconnecting a keyboard, turn off the power of this device.

2.5.2 Description of an External Keyboard

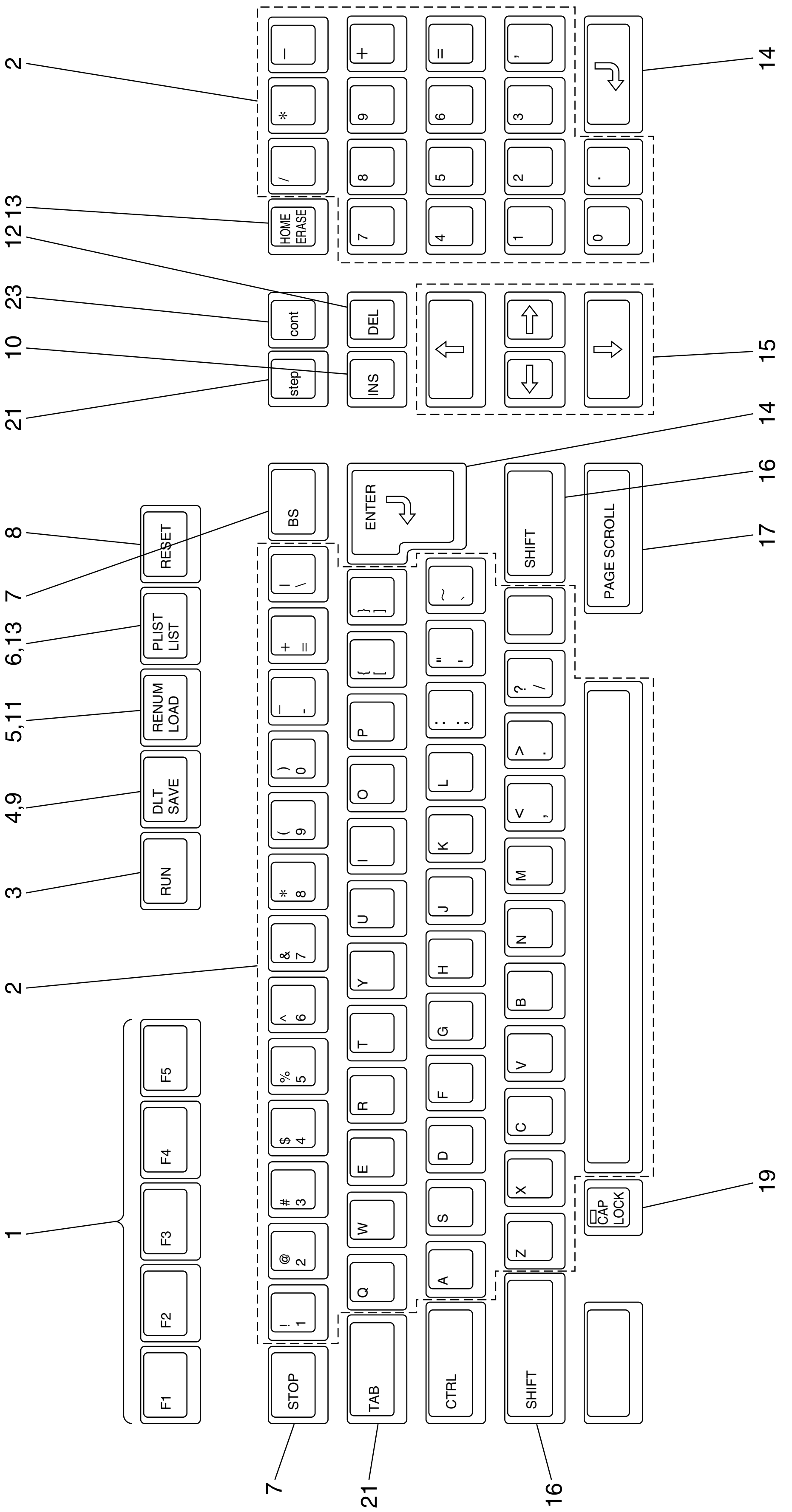
Figure 2-1 shows the key arrangement of an external keyboard.

No.	Key	Function description
1	F1 F2 F3 F4 F5	Keys that can be referenced on the program and correspond to the system variables EX1, EX2, EX3, EX4 and EX5, respectively. Pressing a key will change the variable data to 0 and 1 each time.
2		Inputs a command, a program and data.
3	RUN	Starts executing a program from the first line.
4	SAVE	Displays on screen the SAVE character string of the save command.
5	LOAD	Displays on screen the LOAD character string of the load command.
6	PLIST	Displays on screen the file name and size stored in the storage device (internal memory, FD or PMC).
7	STOP	Suspends executing a program but does not initialize the variables.
8	RESET	Stops executing a program and initializes the variables.
9	DLT	Displays on screen the DELETE character string of the delete command (for deleting a program).
10	INS	Changes the cursor to ◀ (insert cursor) and inserts a character, numeric, or special symbol at the insert cursor when they are pressed. Press the INS key again to change the cursor back to a normal one.
11	RENUM	Displays on screen the RENUM character string of the renumber command (for sorting programs).

No.	Key	Function description
12	DEL	Deletes a character at the cursor position.
13	LIST	Displays on screen the LIST character string of the list command.
14	⏎	Ends entering a command, or one line of a program or data.
15	⇩	Moves the cursor up and down and to the right and left. Pressing the SHIFT and arrow keys will move the cursor to the edge of screen.
16	SHIFT	With the [SHIFT] key down, press the character, numeric or special symbol key to input an upper-case alphabetical character (or lower-case alphabetical character if [CAP LOCK] is turned on) or a special symbol in the upper row.
17	PAGE SCROLL	Displays on screen a program page by page.
18	BS	Deletes a character just before the cursor.
19	CAP LOCK	If [CAP LOCK] is on, the lamp will light up, and pressing a character key will input an upper-case alphabetical character. If [CAP LOCK] is pressed again and turned off, the lamp will go off, and pressing a character key will input a lower-case alphabetical character. Press this key only after the PTA is turned on.
20	HOME ERASE	Clears the screen and displays the cursor at the home position (upper left corner of the screen).
21	TAB	Inputs a fixed-length space.
22	STEP	Executes a program line by line.
23	CONT	Restarts executing a suspended program.



Key Arrangement of an External Keyboard for the PTA



Key Arrangement of an External Keyboard for ASCII, MC3306A

2.6 Handling Storage Media (internal memory, FD and PMC)

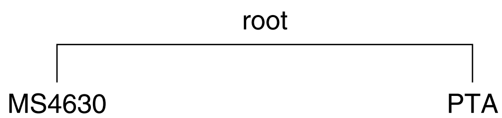
This device has standard internal memory and a 3.5" FD as the storage media. The plug-in memory card (PMC) is also optionally available.

2.6.1 Formatting a Medium

A medium, when used for the first time, must be formatted (to the MS-DOS format). For this purpose, use the menu called up from the SAVE/RECALL key on the front panel. Specify a drive according to the medium to be formatted.

2.6.2 Creating a Directory

Execute the media formatting function. The medium will be formatted and the directories "MS4630" and "PTA" will be created under the root directory as follows:



(1) "MS4630" directory

Under this directory, the data files (waveform data, measurement conditions, calibration data, etc.) to be handled by SAVE/RECALL of the main frame function will be created.

(2) "PTA" directory

Under this directory, the PTA program files, PTA data files, and startup registration files to be handled by the PTA will be created.
Unless the "PTA" subdirectory exists, the PTA files such as the above cannot be saved.

2.6.3 Exchanging the media for MS3401*

The following shows the precautions for using the PMC formatted by the Anritsu's old-model network analyzer (hereafter, old PMC format).

- This device can read the PTA program files and data files saved in the old PMC format. However, this device cannot read the data files saved by the main frame function (waveform data, measurement conditions, calibration data, etc.).
- This device cannot save the data in the PMC in the old PMC format.
- This device cannot use the startup function registered in the old PMC format.

Section 3 PTL Commands

Section 3 PTL Commands

The Personal Test Language (PTL) commands consist of those for editing and executing a PTA program and managing a file as follows:

- PTL commands
  - Editing commands (program input)
    - `AUTO`
    - `PCOPY`
    - `DELETE`
    - `RENUM`
    - `LIST`
    - `LISTG`
    - `SAVEG`
    - `LOADG`
    - `PAGE SCROLL`
    - `ROLL UP/DOWN`
    - `PMEMO`
  - Execution control commands
    - `RUN`
    - `CONT`
    - `STEP`
    - `RESET`
    - `STOP`
  - File management commands
    - `SAVE`
    - `LOAD`
    - `OVERLAY`
    - `PDEL`
    - `PLIST`
    - `STARTP`
    - `CANCEL`

3.1 PROGRAM INPUT COMMANDS

(1) Function

A statement entered with a line number is stored in the program area as a PTA program. A statement with a different line number than those already entered will be added or inserted to the entered ones. A statement with the same number as one of those already entered will be overwritten over the entered one.

(2) Format

Line number statement

Integer constant between 1 and 65535 inclusive

Note:

If you input more than 110 characters per line including a line number and execute the RENUM command, executing the LIST command thereafter may not display this program line.
For the RENUM command, see "RENUM Command" in CHAPTER 3.

3.2 LINE NUMBER AUTOMATIC UPDATING COMMAND

(1) Function

The AUTO command automatically displays and updates line numbers.
If this command is executed, the system will display the specified <Start line number> and waits for the statement input. Thereafter, enter a statement and the RETURN key to store the statement into the program area. The line numbers will then be updated for the increment specified in <Increment>. If <Start line number> and <Increment> are omitted, 10 will be given as the default value.
Press the RESET key to terminate the AUTO command.

(2) Format

```
AUTO  [Operand 1] [, Operand 2]
           |            |
  Start line number   Increment
```

1.	`AUTO`	Updates the line numbers every 10 steps from the 10th line.
2.	`AUTO 100`	Updates the line numbers every 10 steps from the 100th line.
3.	`AUTO, 1`	Updates the line numbers every step from the 10th line.
4.	`AUTO 100, 5`	Updates the line numbers every 5 steps from the 100th line.

Note:

If, while entering a program after defining the start line number and the increment, you rewrite an updated line number, the line numbers will be updated for the increment specified in <Increment>, starting from the rewritten line number.

3.3 PCOPY COMMAND

(1) Function

The PCOPY command copies a statement.

This command copies the portion between the specified <Source start line number> and <Source end line number> at <New start line number> using the increment specified in <Increment>. If <Increment> is omitted, 10 will be given as the default value.

(2) Format

PCOPY	Operand 1,	[Operand 2],	Operand 3,	Operand 4
	New start line number	Increment	Source start line number	Source end line number

1. `PCOPY 100,,10,30` Copies the statements from the 10th to 30th line every 10 steps from the 100th line.
2. `PCOPY 100,5,10,30` Copies the statements from the 10th to 30th line every 5 steps from the 100th line.

Notes:

- If, while defining operands, the new line numbers of the statements to be copied repeat or overlap the line numbers of the existing statements, an error F101 will occur.

- If one line consists of more than 110 characters and PCOPY is executed, executing the LIST command thereafter may not display such a program line.

3.4 DELETE COMMAND

(1) Function

The DELETE command deletes a whole program or a part of a program.

(2) Format

```
DELETE [Operand 1] [,] [Operand 2]
                    |
        Operand 1 ≤ Operand 2
```

1. `DELETE` Deletes all the lines of a program. Initializes the variable values.
2. `DELETE 1ØØ` Deletes the 100th line.
3. `DELETE 1ØØ,` Deletes the portion from the 100th to the last line.
4. `DELETE ,5ØØ` Deletes the portion from the first to the 500th line.
5. `DELETE 1ØØ, 5ØØ` Deletes the portion from the 100th to the 500th line.

- To delete one line with the specified line number (only), enter:
  Line number [RETURN]

3.5 RENUM COMMAND

(1) Function

The RENUM command sorts (renumbers) the line numbers of a program.
If <Increment> and <New line number> are omitted, 10 will be given as the default value.

(2) Format

```
RENUM [Operand 1] [,] [Operand 2] [,] [Operand 3] [,] [Operand 4]
```

[Operand 1]: New Start line number, [Operand 2]: Increment, [Operand 3]: Start line number, [Operand 4]: End line number

1.	`RENUM`	Renumbers a whole program using the increment of 10 with the new start line number as 10.
2.	`RENUM 100`	Renumbers a whole program using the increment of 10 with the new start line number as 100.
3.	`RENUM 100,5`	Renumbers a whole program using the increment of 5 with the new start line number as 100.
4.	`RENUM 100,,50`	Renumbers the portion from the 50th line using the increment of 10 with the new start line number as 100.
5.	`RENUM 100,5,50`	Renumbers the portion from the 50th line using the increment of 5 with the new start line number as 100.
6.	`RENUM 100,,,150`	Renumbers the portion from the 10th to the 150th line using the increment of 10 with the new start line number as 100.
7.	`RENUM 100,,5,150`	Renumbers the portion from the 50th to the 150th line using the increment of 10 with the new start line number as 100.
8.	`RENUM 100,5,,150`	Renumbers the portion from the 10th to the 150th line using the increment of 5 with the new start line number as 100.
9.	`RENUM 100,5,50,150`	Renumbers the portion from the 50th to the 150th line using the increment of 5 with the new start line number as 100.

Notes:

- A label can be used for Operands 1, 3, and 4.
- If Operand 1 is smaller than Operand 4 and a program with a smaller line number than Operand 4 exists, an error F101 will occur.
- If one line consists of more than 110 characters, executing RENUM may increase the number of digits of the line number, causing the program to extend over two lines or more. In this case, executing the LIST command will display this line and cause error F20.

3.6 LIST COMMAND

(1) Function

The LIST command outputs to the screen a whole program or a part of a program.

(2) Format

```
LIST [Operand 1] [,] [Operand 2]
```

Operand 1: Output start line number
Operand 2: Output end line number

Operand1 ≤ Operand 2

1. `LIST` Outputs a whole program.
2. `LIST 1ØØ` Outputs the 100th line.
3. `LIST 1ØØ,` Outputs the portion from the 100th to the last line.
4. `LIST,5ØØ` Outputs the portion from the 1st to the 500th line.
5. `LIST 1ØØ,5ØØ` Outputs the portion from the 100th to the 500th line.

Note:

A label can be used for Operands 1 and 2.

3.7 LISTG COMMAND

(1) Function

The LISTG command outputs the PTA program to a printer connected to the GPIB/RS-232C/parallel (Centronics) interface of the specified port.

(2) Format

```
LISTG  Address [ [,] [Operand 1] [,] [Operand 2] ]
```

Address: Printer address between 0 and 30 inclusive

Use Operands 1 and 2 in the same way for the LIST command.

Notes:

- This command is enabled only when the specified port is a controller port. For information on specifying the port, see CHAPTER 7, "EXTERNAL INTERFACE FOR THE PTA".
- You must specify an address as a matter of form although it is not significant if a program is output to the RS-232C or parallel (Centronics) interface.

3.8 SAVEG COMMAND

(1) Function

The SAVEG command transfers the PTA program on this device to another network analyzer connected to the GPIB/RS-232C interface of the specified port.

(2) Format

```
SAVEG  Address
          |
       0 to 30
```

Notes:

- This command is enabled only when the specified port is a controller port. For information on specifying the port, see CHAPTER 7, “EXTERNAL INTERFACE FOR THE PTA”.
- You must specify an address as a matter of form although it is not significant if a program is output to the RS-232C interface.
- The devices (network analyzers) that can execute a PTA program are MS3401*, MS3606* and this device.

3.9 LOADG COMMAND

(1) Function

This command loads the PTA program from another network analyzer located on GPIB (RS-232C) on the specified port to the device.

(2) Format

```
LOADG  Address
          |
        0 to 30
```

Notes:

- This command is enabled only when the specified port is a controller port. For information on specifying the port, see CHAPTER 7, "EXTERNAL INTERFACE FOR THE PTA".
- You must specify an address as a matter of form although it is not significant if a program is output to the RS-232C interface.
- The devices (network analyzers) that can execute a PTA program are MS3401* and MS3606*.
- The MS3401* and MS3606* add a string "END" at the end of a program that they output. Since this device cannot interpret this string, an error will result. However, the transferred program remain unaffected.

3.10 PAGE SCROLL COMMAND

(1) Function

The PAGE SCROLL command displays a program page by page.
First use the LIST command to display a program. Execute the PAGE SCROLL command for the first time to stop LIST. Then, use the PAGE SCROLL repetitiously to display the program page by page until the last line.

(2) Format

PAGE SCROLL key

SHIFT + PAGE SCROLL keys

Notes:

- Press the SHIFT and PAGE SCROLL keys to execute PAGE SCROLL in the reverse direction.
- The PAGE SCROLL command is enabled only if a program is already displayed using the LIST command.

3.11 ROLL UP/DOWN COMMANDS

(1) Function

The ROLL UP/DOWN commands scroll forward and backward the screen line by line if any program line is displayed using the LIST command.

(2) Format

CTRL + J keys → ROLL UP

CTRL + K keys → ROLL DOWN

Note:

This command is disabled while the AUTO command is executed, after the RUN command is executed, and when the screen is cleared after the HOME ERASE key is entered.

3.12 PMEMO COMMAND

(1) Function

This command outputs, on the screen, the size of memory used by the program area where the PTA program is stored as well as the size of memory required to store data into the storage device (internal memory, FD or PMC).

(2) Format

```
PMEMO
```

(3) Output

1.	Used memory size :	***bytes
2.	PTA program	***bytes
3.	LIB programs	***bytes
4.	Variables	***bytes
5.	Unused memory size :	***bytes

File size :

6.	PTA program	***bytes (ASCII)
7.		***bytes (BINARY)
8.	LIB programs	***bytes (BINARY)

1: Used memory size in the PTA program memory

2: Program size in "1."

3: Not used in this system

4: Not used in this system

5: Unused memory size in the PTA program memory

6: File size (estimated value) required to save a PTA program on the storage device in the ASCII format

7: File size (estimated value) required to save a PTA program on the storage device in the binary format

8: Not used in this system, but secured for this fixed size.

- The file sizes shown in 6. and 7. are somewhat different from those after the files are saved.
- File sizes will be larger in the binary format than in the ASCII format. On the other hand, it will take less time to save or load a file in the binary format.

3.13 IMMEDIATE EXECUTION COMMAND

(1) Function

Enter a statement without a line number and press the ⏎ key to execute the statement immediately. However, you cannot immediately execute a statement that cannot be executed by itself, such as GOTO, GOSUB, RETURN, RETMAIN, IF, FOR, NEXT, DATA, RDATA, RESTORE, and CHAIN.
For details of statements, see CHAPTER 4, “PTL”.

(2) Format

Statement

3.14 RUN COMMAND

(1) Function

The RUN command starts executing a PTA program. The PTA program will be terminated if the STOP statement is executed, if an error occurs, or if RESET is executed.

(2) Format

[RUN] key
RUN [[Operand 1] [, Operand 2]]

Start line number Suspend line number

1. `RUN` Starts executing a program from the beginning of a line.
2. `RUN 1ØØ` Starts executing a program from the 100th line.
3. `RUN ,5ØØ` Starts executing a program from the first line and suspends at the 500th line.
4. `RUN 1ØØ,5ØØ` Starts executing a program from the 100th line and suspends at the 500th line.

Note:

Executing the RUN command will not initialize the variable values.

3.15 STOP COMMAND

(1) Function

The STOP command stops executing a PTA program being executed.

(2) Format

[STOP] key

3.16 CONT COMMAND

(1) Function

The CONT command restarts executing a PTA program being suspended. This command is enabled only if the program is suspended after the RUN or STEP command is executed.

(2) Format

[CONT] key

CONT [Operand]

1. CONT	Restarts executing a program from the line after the one where it was suspended.
2. CONT 1ØØØ	Restarts executing a program from the line after the one where it was suspended, and suspends the execution at the 1,000th line.

3.17 RESET COMMAND

(1) Function

The REST command stops executing a command or a PTA program.

(2) Format

[REST] key

(3) Initialization

- Clears the system variables EX1, EX2, EX3, EX4 and EX5.
- Clears the user-defined variables. Does not clear the system variables.
- Initializes the PTA parallel I/O port.
- Cancels the OVERLAY status (where a program can be loaded by overwriting).

3.18 SAVE COMMAND

(1) Function

The SAVE command saves a PTA program to the storage device (internal memory, FD or PMC). In this case, the file size of the PTA program must be smaller than the unused memory size of the storage device.
Execute the PMEMO command to output to the screen the file size of a PTA program, and the PLIST command to output the unused memory size of the storage device.

(2) Format

```
SAVE  PTAprogram name [.Attribute] [, Operand 1] [, Operand 2]
             |              |            |             |
             |          PTA or IMG  Start line number  End line number
             |
Alphanumeric string consisting of six characters or
less starting with an upper-case alphabetic character
```

Notes:

- A file that was opened using CALL OPNI or OPNO "% File name" will be closed.
- A label can be used for Operands 1 and 2.
- Check that the storage device is formatted before saving a program. An unused the storage device must be formatted before a program can be saved in it. The PTA program file will be created in the "PTA" subdirectory in the medium.
- The program will be saved as an ASCII file if PTA is specified for the attribute. The program will be saved as a binary file if IMG is specified for the attribute. Saving the program as a binary file will shorten the time required to load it. If the attribute is omitted, the program will be saved with .PTA automatically added.
- Select a storage device in the "DRIVE" menu called up by pressing the SAVE/RECALL key of the main frame panel.

3.19 LOAD COMMAND

(1) Function

The LOAD command loads a PTA program from a storage device (internal memory, FD or PMC) to the program area in the main frame. If another PTA program is already stored in the program area, it will be replaced with the new one unless OVERLAY is executed.

(2) Format

```
LOAD  PTA  program name [.Attribute]
                 |               |
                 |           PTA or IMG
Alphanumeric string consisting of six characters or
less starting with an alphanumeric character
```

Notes:

- A file that was opened using CALL OPNI or OPNO "% File name" will be closed.
- If you execute RESET while loading a program, only a part of the program will be loaded.
- The program area (memory) of this device is not battery-backed. Turning off the power switch of this device will cause the content of the program area to be lost.
- Select a storage device drive in the "DRIVE" menu called up by pressing the SAVE/RECALL key of the main frame panel.

3.20 OVERLAY COMMAND

(1) Function

The OVERLAY command specifies, when the LOAD command is executed, to overwrite a PTA program that existed before the load.

(2) Format

```
OVERLAY
```

Note:

This specification continues to be valid until the RESET command is executed.

3.21 PDEL COMMAND

(1) Function

The PDEL command deletes a PTA program from a storage device (internal memory, FD or PMC).

(2) Format

```
PDEL  PTA  program name [.Attribute]
                          |
                     PTA or IMG
```

Notes:

- "% File name" (Data file) cannot be deleted using the PDEL command (see "OPNI, OPNO and FDEL Subroutines" in CHAPTER 5.)
- A file that was opened using CALL OPNI or OPNO "% File name" will be closed.
- If the attribute is omitted, PTA will be automatically added in the process.
- Select a storage device. drive in the "DRIVE" menu called up by pressing the SAVE/RECALL key of the main frame panel.

3.22 PLIST COMMAND

(1) Function

The PLIST command displays the file names, used file sizes, and unused sizes of the files stored on a storage device (internal memory, FD or PMC).

(2) Format

[PLIST] key

(3) Output

The PLIST command displays the data by scrolling the screen page by page. If more files than can be displayed in one page are stored on an FD or a PMC, the data will be displayed as in “1.” below and then scrolled page by page with repetitious executions of the PLIST command. If the files can be displayed in one page, the data will be displayed as in “2.” below.

1. If other pages exist

....	.... bytes	PROG (IMAGE)
%SDAT0.DAT	1024 bytes	DATA
%SDAT2.DAT	1024 bytes	DATA
ABCXYZ. PTA	15808 bytes	PROG (ASCII)
		continue

2. If no other pages exist

BANDLH.PTA	18568 bytes	PROG (ASCII)
RPLLH.IMG	35786 bytes	PROG (IMAGE)
MAXMIN.PTA	27368 bytes	PROG (ASCII)
unused memory size:	89010 bytes	

unused memory size: The unused memory size (in bytes).

Note:

Select a storage device (internal memory, FD or PMC) in the “DRIVE” menu called up by pressing the SAVE/ RECALL key of the main frame panel.

CAUTION ⚠

- **A file that was opened using** CALL $\left\{\begin{matrix}\text{OPNI}\\ \text{OPNO}\end{matrix}\right\}$ **“% File name” will be closed.**
- **The PLIST command displays only the PTA program file created by PTA and data files created in the PTA. It will not display the waveform and measurement parameter files saved using the main frame function. The unused memory sizes may be smaller than expected because of these files.**

3.23 STARTP COMMAND

(1) Function

The STARTP command registers a startup function to turn on the PTA at power-on and loads and executes the specified PTA program.
This function registers the PTA program file located in the storage device.

(2) Format

STARTP   Program name

- At power-on after this registration, the PTA will be turned on and the registered PTA program will be loaded and executed.
- This registration will create a special startup file named “MS4630.bat” in the storage device (not displayed using the PLIST command).
- Even after this registration, the startup function will not be executed if:
  - No storage medium is mounted in the selected drive at power-on.
  - No program having the registered PTA program name exists in the storage medium mounted in the selected drive.

Note:

Select a storage device drive in the “DRIVE” menu called up by pressing the SAVE/RECALL key of the main frame panel.

3.24 CANCEL COMMAND

(1) Function

The CANCEL command cancels the registration of the startup function.

(2) Format

`CANCEL` : Canceling the registration of the PTA program file in the storage device

Note:

Select a storage device drive in the "DRIVE" menu called up by pressing the SAVE/RECALL key of the main frame panel.

Section 4 PTL

The Personal Test Language (PTL) is similar to the BASIC language and consists of basic statements and extended PTL such as system variables, system subroutines, and GPIB statements.

4.1 Components of A Statement

4.1.1 Line Number

(1) Function

A line number is written before a statement and used as an index when the program is executed.

(2) Format

Numeric string

Integer constant between 1 and 65535 inclusive

4.1.2 Constants

(1) Function

A constant represents a specific numeric, character string or bit string.

(2) Format

(a) Numeric constant

[–]
Numeric string
[.Numeric string]
[E [–] Numeric string]

Sign	Integer part	Fractional part	Exponent
	Mantissa		

(A mantissa is max. 15 digits. An exponent is between 10^{308} and 10^{-307} inclusive.)
To an integer numeric variable, assign a numeric constant between
–32768 to +32767 inclusive.

(b) Character constant

"Character string"

1 to 255 characters, enclosed between " and "

Note:

When entering a program, one line will consist of max. two lines on the screen, and you can only enter as many characters as can be entered in this range.

(c) Bit constant

- Hexadecimal constant
  $Hexadecimal notation

  0 to FF

- Binary constant
  $Binary notation

  0 to 11111111

(3) Example

(a) Numeric constant

`1`

`-12.3`

`13E3` .... Same as 12000

`-Ø.12E-3` .... Same as –0.00012

(b) Character constant

`"Who are you?"`

(c) Bit constant

`$F` .... Same as #1111 (binary) or 15 (decimal).

`#ØØØ11Ø1Ø` .... Same as $1A (hexadecimal) or 26 (decimal).

4.1.3 Variables

The variables consist of simple, array and system variables. For the system variables, see "System Variables" in CHAPTER 5.

(1) Simple variables

Simple variables consist of numeric, character string and bit string variables that have a variable name of 8 characters or less (including %, $ and #) starting with an upper-case alphabetical character.

- Real number numeric variable name : Upper-case alphabetical character [Alphanumeric [Alphanumeric]] ——— ABCD0123
- Integer numeric variable name : Upper-case alphabetical character [Alphanumeric [Alphanumeric]]% ——— A%
- Character string variable name : Upper-case alphabetical character [Alphanumeric [Alphanumeric]]$ ——— ABC$
- Bit string variable name : Upper-case alphabetical character [Alphanumeric [Alphanumeric]]# ——— A#

(2) Array variables

Array variables are variables declared as an array in the DIM statement. Some system variables are handled as array variables. The following shows the format of an array variable.

- Array variable : Variable (Numeric constant or numeric variable)

Range of a subscript
- For a numeric array variable 0 to 1023
- For a character array variable 0 to 254
- For a bit array variable 0 to 7

CAUTION ⚠

- **A subscript in an array variable is between 0 and the array size minus one inclusive.**
- **If a subscript is a real number in an array variable, the fractional part of the subscript will be rounded off.**
- **Max. 256 variables (excluding system variables) can be used.**
- **An already registered symbol such as a command, statement, function or system variable cannot be used as a user-defined variable.**

4.1.4 Multiple Statements

Using “&” between statements, you can enter multiple statements in the same line. You can also enter a program over two lines. You can enter any number of statements as long as they fit within two lines.

Example:
```
10 FOR I=0 TO 10 & A=I*I & PRINT A & NEXT I
20 STOP
```

4.1.5 Functions

The PTL functions consist of basic functions (i.e. arithmetic, bool, statistical, and character string functions) and other dedicated functions.
The following shows the basic functions.

(1) Arithmetic functions

Function name	Function	Parameter	
Sine	`SIN (X)`	X is in degrees.	X: Numeric constant or variable.
Cosine	`COS (X)`		
Tangent	`TAN (X)`	$X \neq 90(2n+1)$, n = Any integer	
Arc sine	`ASN (X)`	$\lvert X \rvert \leq 1$	
Arc cosine	`ACS (X)`		
Arc tangent	`ATN (X)`		
Natural logarithm	`LN (X)`	$X > 0$	
Common logarithm	`LOG (X)`		
Exponent	`EXP (X)`		
Square root	`SQR (X)`	$X \geq 0$	
Absolute value	`ABS (X)`		
Sign	`SGN (X)`	If X > 0, SGIN(X) = 1 If X < 0, SGIN(X) = –1 If X = 0, SGIN(X) = 0	
Integer value	`INT (X)`	X : Numeric constant variable (An integer not exceeding X will be returned.)	
Rounding off	`ROUND (X [,N] )`	X : Numeric constant variable N : Numeric constant variable (If omitted, N=0) (X will be rounded off to the N-th decimal place.)	
Function to obtain a quotient and a remainder	`Q=DIV (R,S,D)`	Q : Numeric variable ... the quotient will be stored. R : Numeric variable ... the remainder will be stored. S : Integer variable ... the dividend will be stored. D : Integer variable ... the divisor will be stored.	
Function to obtain the integer and fractional parts from a real number value.	`I=FIX (S,D)`	I : Integer variable ... the integer part will be stored. S : Real number variable ... the original real number will be stored. D : Real number variable ... the fractional part will be stored.	

(2) Boolean functions

Function name	Function	Parameter
Logical opposite	`NOT (X)`	X and Y is a bit or integer constant, variable and hexadecimal constant.
Logical product	`AND (X,Y)`	
Logical sum	`OR (X,Y)`	
Exclusive logical sum	`EOR (X,Y)`	

(3) Statistical functions

Function name	Function	Parameter
Function to obtain the maximum value	`MX=max (S)`	S: Variable with the one-dimensional array definition
Function to obtain the minimum value	`MN=min (S)`	MX: The maximum value will be stored.
Function to obtain the sum value	`SM=sum (S)`	MN: The minimum value will be stored.
Function to obtain the mean value	`MS=mean (S)`	SM: The sum value will be stored.
Function to obtain the variance value	`VR=var (S)`	MS: The mean value will be stored.
Function to obtain all the values shown above	`VR=sta(S,MX,MN, SM,MS)`	VR: The variance value will be stored. $\text{Variance} = \frac{\Sigma\,(x-x)^2}{\text{Number of samples}}$

CAUTION ⚠

- **On the left side are the numeric variables in which the obtained values can be stored.**
  The parameter S, if defined in one dimension, will be valid even with only one component. To specify all the components on which to perform the statistical processing, no subscript is required while being entered. If a subscript is added, only that component will be processed.
- **In this device, a floating-point numeric processor performs all the function calculations and other operations. The result of these calculations and operations may have errors due to borrows and rounding errors.**

(4) Character string function

(a) Character string function that converts a numeric to a character (string) or vice versa

1. ASC (character constant variable)

   Gives a character code of the first character in the character string.

2. CHR$ (constant variable)

   Gives a character having the character code of the parameter value. Gives the character itself if it is a character type. Specify a parameter between 0 and 225 inclusive.

3. STRING$ (constant variable, character constant variable, or numeric constant variable)

   Gives characters having the character code of the first or numeric in a character string specified in the second parameter. Gives as many such characters as specified in the first parameter (between 0 and 255 inclusive).
   See CHR$ ( ).

4. HEX$ (Numeric constant variable 1[, Numeric constant variable 2])

   Gives a numeric in decimal in the first parameter as a character string in hexadecimal having the number of digits in the second parameter.
   If the value in the first parameter exceeds the range between -2^{31} and 2^{32} - 1 inclusive, an error will occur. If the specification of the second parameter exceeds 8 digits, an error will occur. If omitted, the character string will be variable length.

5. OCT$ (Constant variable)

   Gives the parameter value as a character string in octal. If the value exceeds the range between -32768 and 32767, an error will occur.

6. BIN$ (Numeric constant variable 1 [, Numeric constant variable 2])

Gives a numeric in decimal in the first parameter as a character string in binary having the number of digits in the second parameter.
If the value in the first parameter exceeds the range between -2^{31} and $2^{32} - 1$ inclusive, an error will occur. If the specification of the second parameter exceeds 32 digits, an error will occur. If omitted, the character string will be variable length.

7. CVI (Character constant variable of two characters or more)

Gives a character string converted to an integer numeric. If the character string has more than two digits, the exceeding part will be ignored. If the character string has less than two digits, an error will occur.

8. CVD (Character constant variable of eight characters or more)

Gives a character string converted to a double precision integer numeric. If the character string has more than eight digits, the exceeding part will be ignored. If the character string has less than eight digits, an error will occur.

9. MKI$ (Integer constant variable)

Directly converts the value expressed internally (in binary) in a numeric to a character code.
This is opposite to the processing described for CVI ( ).

10. MKD$ (Double precision integer constant variable)

Directly converts the value expressed internally (in binary) in a numeric to a character code.
This is opposite to the processing described for CVD ( ).

11. VAL (Character variable, Numeric variable 1, Numeric variable 2)

Assuming the numeric specified in the second parameter as m, and in the third parameter as n, this function retrieves the numeric character string from among the m-th to n-th characters from the beginning of a character string, ignoring non-numeric codes, and coverts the character string to a double precision real number numeric.
The parameter m or n can be omitted. If m is omitted, the portion from the beginning of the character string will be processed. If n is omitted, the portion to the end of the character string will be processed. If no numeric is found, an error will occur.

12. BVAL (Character constant variable)

Converts the parameter expressed in binary to a numeric in decimal without a sign. If the parameter exceeds 32 bits, an error will occur. A character except “0” and “1” will be ignored.

13. HVAL (Character constant variable)

Converts the parameter expressed in hexadecimal to a numeric in decimal without a sign. If the parameter exceeds 32 bits (8 bits), an error will occur. A character except “0” to “9” and “A” to “F” will be ignored.

14. CHR (Numeric constant variable)

Out of the numeric specified in the parameter, this function gives the same character string as will be displayed using the PRINT statement.

15. STR$ (Numeric constant variable)

Performs the same processing as described for the function CHR ( ).

(b) Functions that search for a character string, etc.

1. INSTR ([Numeric constant variable,] Character constant variable 1, Character constant variable 2)

   Searches String 1 for String 2 and returns a value representing the location if found, or the value 0 otherwise. The first parameter, if being a numeric, indicates the location to start the search or, if omitted, specifies to start the search from the beginning. Specify a value between 1 and 225 inclusive.

2. LEFT$ (Character variable, Numeric constant variable)

   Gives as many character strings as specified in the parameter from the left end of the character string. If more character strings than the total number of character strings are specified, this function gives all the character strings. Specify a value between 0 and 225 inclusive. If 0 is specified, a blank character string will be given.

3. MID$ (Character variable, Numeric constant variable 1, Numeric constant variable 2)

   Assuming the numeric specified in the second parameter as m, and in the third parameter as n, this function gives the character strings from the m-th to n-th characters from the beginning of a character string. Specify a value between 1 and 256 inclusive for m, and between 1 and 255 inclusive for n. If m exceeds the character string length, a blank character string will be given.

4. RIGHT$ (Character constant variable, Numeric constant variable)

   Performs the same processing as LEFT$ ( ) starting from the right end of a character string. Specify a value between 0 and 225 inclusive. However, the order of character strings will not be reversed.

5. LEN (Character constant variable)

   Gives the number of characters in the character string including the character codes from 0 to $1F.
   The size declared in the array will be given.

6. SLEN (Character constant variable)

   Similar to LEN ( ) and gives the number of characters in the character string, but excluding the spaces at the end of the character string.

7. SGET (Character constant variable)

   Gives a significant character string excluding the spaces at the end of the character string.

(5) Dedicated functions

Function name	Function	Parameter
Reading an error code and an error occurrence	`V=ERRREAD (m)`	m 0 : Error code : Line number where error occurred
Reading event factors	`A#=STATUS (m)`	m 0 : Factor 0 1 : Factor 1 2 : Factor 2 3 : Factor 3
Reading year, month and day/hour, minute and second	`A$=DTREAD$ (m)`	m 0 : Year, month and day (YY-MM-DD) 1 : Hour, minute and second (HH:MM:SS)
Generating a random number (more than 0 and less than 1)	`RND (m)`	m : Any value

CAUTION ⚠

- **ERRREAD (m) can be used only during an error interrupt. For details of an error interrupt, see "ON ERROR Statement" in CHAPTER 4.**
- **STATUS (m) can be used only during an event interrupt. For details of an event interrupt, see "ENABLE EVENT Statement" in CHAPTER 4.**
- **For m, use a numeric constant or a numeric variable.**
- **Pseudo-random numbers generated using RND (m) will appear in the same order every time the command is run. For information on changing the order, see "RNDMIZE Statement" in CHAPTER 4.**

4.1.6 Arithmetic Operators

(1) Function

These operators represent the four basic arithmetic operations and the exponential operation.

(2) Format

= .... Represents assignment.
\+ .... Represents addition.
− .... Represents deduction.
* .... Represents multiplication.
/ .... Represents division.
! .... Represents exponential operation.
( ) .... Represents priority (Equation inside brackets are calculated first.)

(3) Priority

Table 4-1 shows the priority.

Table 4-1 Priority of Arithmetic Operations

Priority	Arithmetic Operator
High	!
↕	* /
	+ −
Low	=

CAUTION ⚠

- **Bits and characters cannot be calculated.**
- **X!Y can be calculated if X is a negative number and Y is a positive number.**
- **If variables of different types exist on the right side of an equation, an overflow or underflow error may occur in the operation.**
- **In the division of numerics and variables, the resolution of the dividend will be that of the solution.**

(4) Example

```
A$="abc"
C=(D+1ØØ)/E
J=((K+1)*1Ø-M)*1Ø
```

4.1.7 Relational Operators

(1) Function

These operators represent the relational operations.

(2) Format

= .... Represents =.
>< or <> .... Represents ≠.
> .... Represents >.
<= or =< .... Represents ≤.
< .... Represents <.
>= or => .... Represents ≥.

(3) Comparing character strings

To compare character strings for size, the system will compare the significant character strings excluding the spaces at the end of character strings on both sides of an operator.

- If two character strings have the same length, the system will compare them starting from the first character and decide as smaller the character string with a smaller character code value for the first non-matching character that appears.
  <Example> ABC is smaller than ABXD.

- If two character strings have different lengths, the system will compare them for the corresponding length range. If the character strings are found to be the same, then the system will decide the shorter one as smaller.
  <Example> ABX is larger than ABCD.
  ABC is smaller than ABCD.

- The character string with the length 0 is the smallest one.
  <Example> When DIM A$(10) is declared, the length of A$ is 0.

(4) Example

```
IF C=Ø GOTO 1ØØ
IF JKL>=168 STOP
```

4.1.8 Connecting Character Strings (Operator “+”)

(1) Function

Use the operator “+” to connect character strings.

(2) Format

$$\left\{\begin{array}{l}\text{Character string constant}\\\text{Character variable}\\\text{Character string function}\end{array}\right\} + \left\{\begin{array}{l}\text{Character string constant}\\\text{Character variable}\\\text{Character string function}\end{array}\right\}$$

Notes:

- This operator can be used only on the right side of the LET statement.
- This operator cannot connect a character and a numeric, a character and a bit, or a bit and a bit.

(3) Example

```
1ØØ A$="ABC"
11Ø B$="DEF"
12Ø A=INATR (A$,"␣") -1
13Ø B=INSTR (B$,"␣") -1
14Ø C$=LEFT$ (A$,A) +LEFT$ (B$,B)
15Ø PRINT "A$=",AS$
16Ø PRINT "B$=",B$
17Ø PRINT ",C$=",C$
```

```
A$=  ABC␣␣␣␣␣␣␣
B$=  DEF␣␣␣␣␣␣␣
C$=  ABCDEF␣␣␣␣
```

Space

CAUTION ⚠

- **The simple character string variable is considered an array-declared variable of 10 characters, unless otherwise specified, and is filled with space when not filled with assigned character string.**
- **Using the above method, only the characters actually stored can be coupled.**

4.1.9 Data format

(1) Function

Represents the format of a character string used in the output. An integer, a real number without an exponent, a real number with an exponent, a binary and a hexadecimal can be used.

(2) Format

- Integer

  :I Number of digits

  (Between 1 and 18 inclusive)
- Real number without an exponent

  :F Total number of digits. Number of digits in fractional part (Total number of digits ≥ Number of digits in fractional part + 3)

  (Between 4 and 20 inclusive)
- Real number with an exponent

  :E Total number of digits. Number of digits in fractional part (Total number of digits ≥ Number of digits in fractional part + 8)

  (Between 9 and 24 inclusive)
- Character string

  :C Number of digits

  (Between 0 and 255 inclusive)
- Binary

  :B Number of digits

  (Between 1 and 8 inclusive)
- Hexadecimal

  :H Number of digits

  (Between 1 and 2 inclusive)

(3) Example

```
PRINT A$ : C3, J : F1Ø.4
```

CAUTION ⚠

- **Specifying the number of digits in a character string as 0 will specify the character string variable as a variable length, and output the data as long as the character string variable actually is.**
- **Specifying a lower-case characters for I, F, E, C, B, H in the above data format will delete the last character space.**
- **For details, see "PRINT" in CHAPTER 4.**

4.1.10 Label

(1) Function

Write a label together with a line number to indirectly specify the jump destination addresses for statements such as GOTO and GOSUB.

(2) Format

Line number ␣ *Label
Line number ␣ *Label ␣ Statement

- A label consists of eight upper-case alphabetical or alphanumeric characters or less with an upper-case alphabetical initial. An asterisk (*) must be added to the beginning of a label.
- If the same label is defined for different line numbers, an error will occur when the program is executed.

(3) Example

```
1Ø  INPUT A
2Ø  IF A=Ø GOSUB *ABC1
3Ø  IF A<>Ø GOSUB *ABC2
4Ø  GOTO 1Ø
1ØØ *ABC1
11Ø PRINT“OK!”
12Ø RETURN
2ØØ *ABC2
21Ø PRINT“NG!”
22Ø RETURN
```

4.2 Basic Statements

4.2.1 Remark (REM Statement)

(1) Function

Gives a remark to a program. This statement does not affect the program execution in any way. To give any words, enclose the words between quotation marks (") to make them a character constant. An apostrophe (') can be used instead of REM. If an apostrophe is used, the words need not be enclosed between quotation marks.

(2) Format

`REM` ["Array of statement components"] or '[Array of statement components]

(3) Example

```
1Ø  REM
2Ø  REM "Compute average"
3Ø  'Compute average
4Ø  A=1ØØ 'Initial set
```

4.2.2 Array Declaration (DIM Statement)

(1) Function

Declares an array. The array must be either one- or two-dimensional and have a size specified in (2) below depending on the variable name type.

(2) Format

DIM Variable name (Array size [, Array size]) [, Variable name (Array size [, Array size]) ...)

Notes:

- An array cannot be redefined for the same variable name.
- An array declaration cannot be made for a variable name defined as independent variable.
- Referencing (reading and writing) an array defined as two-dimensional without a two-dimensional specification will cause error W225.
- Referencing (reading and writing) an array defined as one-dimensional with a two-dimensional specification will cause error W224.
- The number of arrays that can be declared are as follows. Any specification exceeding this range will cause error 203.

Character type .... 1 to 225	Two-dimensional array:	
Bit type .... 1 to 8	1st dimension side	2nd dimension side
Numeric type .... 1 to 1024	1 to 1024	Character type .... 1 to 225
		Bit type .... 1 to 8
		Numeric type .... 1 to 1024

- Because the program area will become insufficient for a numeric type array variable, it is impossible to define 1024 in either the 1st or 2nd dimensions. If you do this, error F 206 occurs. The total of declarable arrays, depending on the free memory capacity, cannot be determined.
- The character is fixed to the 10-character type if no array declaration is made.
- The bit type is fixed to 8 characters unless an array declaration is made.
- Referencing (reading and writing) components without making an array declaration will cause error W224.

(3) Example

```
DIM   CARR(1ØØ),A$(5,12)
DIM   I#(8),ALP$(4Ø)
```

(4) System variables unconditionally declared as array sources

```
XMA(1ØØ1),XMB(1ØØ1),SMA(1ØØ1),SMB(1ØØ1),IMA(1ØØ1),IMB(1ØØ1)
IDA(1ØØ1),IDB(1ØØ1),FQM(1ØØ1),LVM(1ØØ1)
IOA(8),IOB(8),IOC(4),IOD(4),
FI1(1Ø),FO1(1Ø),FI2(3),FO2(4),FI3(13),FO3(8)
FI4(4),FO4(3),FI5(1Ø),FO5(5)
```

Inside the brackets ( ) is the number of components in an array.

4.2.3 Initialization (CLEAR Statement)

(1) Function

Initializes the user-defined variables.

(2) Format

```
CLEAR
```

Note:

Executing the CLEAR statement, like the RESET statement, will initialize the variables, requiring the arrays to be redefined.

4.2.4 Assignment (LET Statement)

(1) Function

Assigns to a variable a constant, a variable, or an operation result from them.

For operators, see “Arithmetic Operators” in CHAPTER 4.

(2) Format

$$\text{[LET] variable= [( ]}\left\{\begin{array}{l}\text{Constant}\\ \text{Variable}\\ \text{Function}\end{array}\right\}\text{[ )]}$$

$$\text{[Arithmetic Operator] [( ]}\left\{\begin{array}{l}\text{Constant}\\ \text{Variable}\\ \text{Function}\end{array}\right\}\text{[ )]}$$

$$\begin{array}{c}+\ -\\ *\ /\\ !\end{array}$$

$$\text{[LET] character variable=}\left\{\begin{array}{l}\text{Character string constant}\\ \text{Character variable}\\ \text{Character string function}\end{array}\right\}+\left\{\begin{array}{l}\text{Character string constant}\\ \text{Character variable}\\ \text{Character string function}\end{array}\right\}+\cdots$$

Notes:

- The bits and characters cannot be calculated.
- If an assignment statement is entered after the IF statement, LET cannot be omitted.

(3) Example

```
LET A=B+C or A=B+C
IF X=Ø LET Y=1Ø
```

4.2.5 Branch (GOTO Statement)

(1)Function

Directs the flow of programs to the specified line number.

(2) Format

`GOTO` Line number or `GOTO` ∗Label

4.2.6 Terminating Execution (STOP Statement)

(1) Function

Displays a execution termination message on the screen and terminates executing a program.
Display the execution termination message as follows.

STOP IN Line number

(2) Format

`STOP`

Note:

Specifying to suspend execution in a line of the STOP statement will not put the device in the suspended state because the program execution will be terminated.

4.2.7 Branch to a Subroutine (GOSUB Statement)

(1) Function

Directs the flow of programs to a subroutine with the specified line number. Executing the RETURN statement at the end of the subroutine will redirect the flow of programs to the line just after the GOSUB statement.

(2) Format

`GOSUB` Line number or `GOSUB` ∗Label

Note:

Calling another subroutine inside a subroutine is called “nesting”. The nesting can be repeated up to ten times in the PTA.

4.2.8 Returning from a Subroutine to a Main Routine (RETMAIN Statement)

(1) Function

During the program execution, the RETMAIN statement allows you to return to the topmost layer routine regardless of the nesting.

(2) Format

```
RETMAIN
```

Note:

Executing the RETMAIN statement in the topmost layer routine, error F213 will occur.

4.2.9 Returning from a Subroutine (RETURN Statement)

(1) Function

Redirects the flow of programs to the line just after the corresponding GOSUB statement.

(2) Format

```
RETURN
```

4.2.10 Decision (IF Statement)

(1) Function

Compares the relationship between constants or variables and, if the result is true, executes the subordinate statement. For relational operators, see “Relational Operators” in CHAPTER 4.

(2) Format

```
   ┌ Constant                       Constant ┐
IF ┤           Relational operator           ├ Statement
   └ Variable                       Variable ┘
                     |
                     =
                     >< or <>
                     >
                     <= or =<
                     <
                     >= or =>
```

Notes:

- You can write any statement as a subordinate statement, including an IF statement.
- The numerics, characters, and bits cannot be compared with each other.
- Omitting LET is not possible when planning to write an assignment statement after the IF statement.

(3) Example

```
IF C=1 GOTO 100
IF ACH$=BCH$ PRINT ACH
IF C<10 IF C>=20 PRINT “ERROR”
IF C<10 LET C=10
```

4.2.11 Starting Iteration (FOR Statement)

(1) Function

Gives initial, end, and increment values to numerics, and repetitiously executes the portion between this statement and the corresponding NEXT statement until the variable becomes the end value. Up to ten layers of nesting can be used for the FOR statement.

(2) Format

$$\text{FOR numeric variable}=\underbrace{\left\{\begin{array}{l}\text{Numeric constant}\\ \\ \text{Numeric variable}\end{array}\right\}}_{\text{Initial value}}\ \text{TO}\ \underbrace{\left\{\begin{array}{l}\text{Numeric constant}\\ \\ \text{Numeric variable}\end{array}\right\}}_{\text{End value}}$$

$$[\text{STEP}\ \underbrace{\left\{\begin{array}{l}\text{Numeric constant}\\ \\ \text{Numeric variable}\end{array}\right\}}_{\text{Step value (1 if omitted)}}]$$

Notes:

- Even if the initial value is larger than the end value, the FOR statement will execute a program at least once.
- There is no limitation where the FOR and NEXT statements should be put in relation to each other. Only the order of executing programs must be correct.

(3) Example

```
FOR C=1 to 1ØØ
FOR T=TB TO TE STEP Ø.1
FOR D=-1 TO -1Ø STEP -1
NEXT D
NEXT T
NEXT C
```

Repeated

4.2.12 Terminating Iteration (NEXT Statement)

(1) Function

This statement, used together with the FOR statement, specifies the end of iteration.

(2) Format

`NEXT` Numeric variable

Same variable as the `FOR` statement

4.2.13 Key Input (INPUT Statement)

(1) Function

Enters the data from the ten-key pad on the front panel. Executing the INPUT statement will display the following message on the screen.

?

Enter data after ? in the same way as for a command input, and press the [ENTER] key in the front panel. Use "," as a separator of data.

(2) Format

`INPUT` ["Displayed character string"] Variable [, Variable ...]

Notes:

- If a real number is entered for an integer variable, the fractional part will be rounded off.
- If the input data is shorter than a character variable, spaces will be added. If longer, the tail end will be ignored.
- A hexadecimal cannot be entered.
- Up to five variables can be entered.
- To enter a comma (,), use the [kHz] key on the front panel.
- To enter a minus (–), use the [+/-] or [MHz] key on the front panel.

(3) Example

```
INPUT "COUNT= ",C   → COUNT=?123
INPUT C,A$,I#       → ?123,Q,1Ø11Ø1
```

4.2.14 Display (PRINT Statement)

(1) Function

Edits the data and displays it on the screen.

The data without a data format will be displayed by showing the significant digits and additional spaces after them.

Table 4-2 shows the data format and the output format.

For the data format, see “Data Format” in CHAPTER 4.

Adding “;” at the end of a line will disable the linefeed.

Table 4-2 Data Format and Output Format

Data format	Output format
I	Integer with zeros suppressed (Example: ␣␣ 123)
F	Integer with zeros suppressed and fraction with zeros suppressed (with sign digit) (Example: ␣␣ 123.45␣)
FP	Integer with zeros suppressed and fraction with zeros suppressed (without sign digit) (Example: ␣ 123.45 ␣)
E	$\left\{\begin{matrix} ␣ \\ - \end{matrix}\right\}$ Fraction E [–] exponent with zeros suppressed (Example: ␣ 123E-2␣␣␣)
C	Character string ... If shorter, data is filled with spaces. If longer, the tail end is ignored.
B/H	Binary or hexadecimal with zeros suppressed (Example: ␣␣ 1011)

(2) Format

$$\texttt{PRINT}\left\{\begin{matrix}\text{Variable [: Data format]}\\ \\ \underset{\text{Constant is displayed as is.}}{\text{Character string constant}}\end{matrix}\right\}\ [,\left\{\begin{matrix}\text{Variable [: Data format]}\\ \\ \text{Character string constant}\end{matrix}\right\}\ \cdots]\ \underset{\text{No linefeed}}{[;]}$$

Notes:

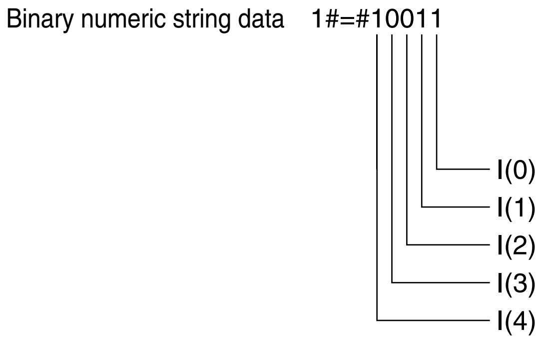
- Up to five variables or constants can be used.
- The numeric that cannot be displayed will be displayed as ***...*.
- The array variables of a character string are as follows:

Character string data A$="ABCDE"

Array variables A(4) A(3) A(2) A(1) A(0)

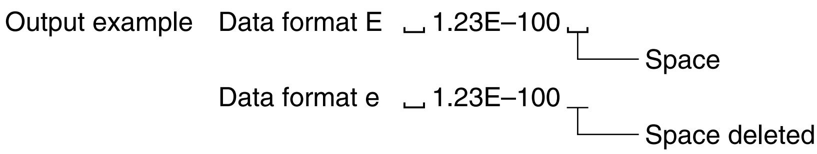
Note:

The array variables of a binary numeric string are as follows:



Note:

Use the data formats i, f, fp, e, c, b, and h (lower case) instead of I, F, FP, E, C, B, and H to delete the spaces at the end of the line.



(3) The following shows the data examples and PRINT output examples.

Table 4-3 shows the output examples of the PRINT statement.

Table 4-3 PRINT Statement Output Examples (1/2)

Data format	Data example	Command	Output
None	T=1234.45	PRINT␣T	123.45␣
	A$="ABCD"	DIM␣A$ (5) PRINT␣A$ PRINT␣A$ (2)	 ABCD␣␣ C␣
	A$=(Ø,)="AB" A$=(1,)="CD" A$=(2,)="EF"	DIM␣A$ (3, 2) PRINT␣A$ (1, Ø) PRINT␣A$ (2,)	 C␣ EF␣
I	T=1234.56	PRINT␣T : I6 PRINT␣T : I4 PRINT␣T : I3	␣␣1234␣ 1234␣ ***␣
F	T=–123.45	PRINT␣T : F6.1 PRINT␣T : F9.2 PRINT␣T : F9.3	–123.4␣ ␣␣–123.45␣ ␣ –123.45Ø␣
	T=123456	PRINT␣T : F9.1 PRINT␣T : F5.1	␣ 123456.Ø␣ *****␣
FP	T=123.45	PRINT␣T : FP6.1 PRINT␣T : FP9.2 PRINT␣T : FP9.3	␣ 123.4␣ ␣␣␣ 123.45␣ ␣␣123.45Ø␣
	T=123456	PRINT␣T : FP9.1 PRINT␣T : FP5.1	␣123456.Ø␣ *****␣
E	T=–123.45	PRINT␣T : E1Ø.2 PRINT␣T : E13.5 PRINT␣T : E15.7	–1.23E2␣␣␣␣ –1.2345␣ F2␣␣␣␣ –1.2345␣␣␣E2␣␣␣␣
	T=–Ø.12E1	PRINT␣T : E9.2	–1.2␣EØ␣␣␣␣
C	A$="F"	PRINT␣A$: C3	F␣␣␣
	A$="ABCDE"	DIM␣AR (5) PRINT␣A$: C7 PRINT␣A$: C3 PRINT␣A$: C5 PRINT␣A$ (3) : C3	 ABCDE␣␣␣ ABC␣ ABCDE␣ D␣␣␣
	A$="ABCDEF"	DIM␣A$ (6) PRINT␣A$ PRINT␣A$ (3)	 ABCDEF␣␣␣ D␣

Table 4-3 PRINT Statement Output Examples (2/2)

Data format	Data example	Command	Output
B	I#=#1	PRINT␣I# : B1 PRINT␣I# : B3	1␣ ØØ1␣
	I#=#1Ø11	DIM␣I# (4) PRINT␣I# : B5 PRINT␣I# : B3 PRINT␣I# (3) : B3 PRINT␣I# (Ø) : B1	 1Ø11␣␣ Ø11␣ 1␣␣␣ 1␣
	I#=#1 I#=#1Ø11	PRINT␣I# DIM␣I# (4) PRINT␣I#	␣␣␣␣␣␣␣␣1␣ 1Ø11␣
	I#=#1ØØ11Ø1Ø	DIM␣I# (8) PRINT␣I# PRINT␣I# (3)	 1ØØ11Ø1Ø␣ 1␣
	I#=#ØØØ1ØØ11	PRINT␣I#	␣␣␣ ␣1ØØ11
H	I#=#1	PRINT␣I# : H1 PRINT␣I# : H2	1␣ ␣1␣
	I#=#1Ø1Ø	DIM␣I# (4) PRINT␣I# : H1 PRINT␣I# : H2	 A␣ A␣␣
	I#=#ØØØØ1Ø1Ø	DIM␣I# (8) PRINT␣I# : H1 PRINT␣I# : H2	 A␣ ␣A␣
	I#=#111Ø1Ø1Ø	DIM␣I# (8) PRINT␣I# : H1 PRINT␣I# : H2 PRINT␣I# (3) : H1 PRINT␣I# (3) : H2 PRINT␣I# (4) : H1 PRINT␣I# (4) : H2	 A␣ EA␣ 1␣ 1␣␣ Ø␣ Ø␣␣
	I#=#ØØ11ØØ	DIM␣I# (6) PRINT␣I# : H2	 ␣C␣
	I#=#11ØØ1Ø	PRINT␣I# : H2	32␣

CAUTION ⚠

The data example with a DIM command indicates one with an array declaration. The data example without a DIM command indicates one without an array declaration.

4.2.15 Reverse Display (PRINTR Statement)

(1) Function

Edits the data and reverse-displays it on the screen.
For other details, see "PRINT Statement" in CHAPTER 4.

(2) Format

$$\texttt{PRINTR}\left\{\begin{array}{l}\text{Variable [: Data format]}\\ \\ \underbrace{\text{Character string constant}}_{\text{Constant is displayed as is.}}\end{array}\right\}\text{ [, }\left\{\begin{array}{l}\text{Variable [: Data format]}\\ \\ \text{Character string constant}\end{array}\right\}\cdots\text{] }\underbrace{\text{[;]}}_{\text{No linefeed}}$$

Notes:

- Only the characters with the character codes 0 to 127 can be displayed in reverse video. The PRINTR statement including other characters than these works in the same way as PRINT and displays the characters in normal video.

- A line including the characters with the character codes 128 to 255 cannot be displayed in reverse video. For a line including these characters, PRINTR works in the same way as PRINT and displays the characters in normal video.

4.2.16 Specifying the Cursor Location (LOCATE Statement)

(1) Function

Specifies the cursor location on the screen (with the upper left of the screen as the home position).

(2) Format

`LOCATE (m,n)`
m → String position (1 to 68)
n → Line location (1 to 30)

Note:

Both m and n are numeric constant variables.

4.2.17 Data Statement (DATA Statement)

(1) Function

Defines the numeric, bit, and character constants that the DATA statement reads.

(2) Format

`DATA` Constant, Constant, ···

Note:

As long as parameters for the DATA statement are written within two lines, any number of parameters can be entered, and constants of different types can be mixed in one statement.

4.2.18 Reading Data (RDATA Statement)

(1)Function

Reads the values from the DATA statement and assigns them to the variables.

(2) Format

`RDATA` Variable, Variable, ···

Notes:

- As long as parameters for the RDATA statement are written within two lines, any number of parameters can be entered, and constants of different types can be mixed in one statement.
- When the format of the variable to be assigned and the format of the definition in the DATA statement cannot be assigned during the reading by the RDATA statement, error W208 will be generated.

4.2.19 Specifying the Data Statement to Be Read (RESTORE Statement)

(1) Function

Specifies the DATA statement to be read by the RDATA statement.

(2) Format

`RESTORE` [Line number or *Label name]

Example:

```
100      RESTORE 1000
110      FOR I=0 TO 10
120      RDATA A (I)
130      NEXT I
             :
1000     DATA 0,1,3,7,9,11,13,17,19,23,29
```

Note:

The parameters of the RESTORE statement can be omitted. If omitted, the first DATA statement will be specified.

4.2.20 Defining the Measurement Parameters (PUT and WRITE 1000 Statements)

(1) Function

Defines the measurement parameters on the main frame from the PTA. The same messages will be used as for defining the parameters using the remote control. These statements can be used to send an inquiry message to the main frame. These statements can be used without any conditions of device/controller of the external interface port.

(2) Format

`PUT` Character constant or Character variable
`WRITE 1000`, Variable or Character constant [, Variable or Character variable]

1. PUT statement
   - Writes in an operand a message of the same format as in the remote control.
   - Only a character constant or a character variable can be written in an operand.
   - Only one constant or variable can be written.
   - The data format cannot be specified.
   - To always define a constant or variable at the fixed location, use this statement to simplify the program.

   Example:

   ```
   PUT "CNF 1ØØ MHZ"
   ```
   → Defines the measurement parameter center frequency to 100 MHz.

   ```
   PUT "CNF?"
   ```
   →Sends a message inquiring for the measurement parameter center frequency.

2. WRITE 1000 statement
   - Writes in an operand a message of the same format as in the remote control.
   - A variable or a character constant can be written in an operand.
   - Up to five constants or variables can be written.
   - If a variable is used, the data format can be specified.
   - Use this statement if you make definitions several times by changing a part of a control message, or use as a setting value what is handled as a variable in the program.

   Example:

   ```
   F=1ØØ
   WRITE 1ØØØ,"CNF ",F,"MHZ"
   ```
   → Defines the measurement parameter center frequency to 100 MHz.

   ```
   WRITE 1ØØØ,"CNF?"
   ```
   → Sends a message inquiring for the measurement parameter center frequency.

4.2.21 Reading the Measurement Parameters and Data (GET, COM and READ 1000 Statements)

(1) Function

Reads the measurement parameters and the measurement results on the main frame from the PTA. The same messages will be used as for defining the parameters using the remote control. These statements can be used without any conditions of device/controller of the external interface port.

(2) Format

`GET` "Inquiry command?", Input variable
`COM` "Inquiry command?">Input variable[, Input variable]
`READ 1ØØØ`, Input variable[, Input variable] or
`READ 1ØØØ`, Input variable[;]

1. GET statement
   - This one statement can both send an inquiry command and read the response data. However, only one inquiry command can be written per statement.
   - Only a character constant or a character variable can be written in an "inquiry command". Only one constant or variable can be written. The data format cannot be specified.
   - The response data will be stored in the input variable. If the response data includes a character, specify a character variable. If the response data includes only numerics (numeric characters), either a numeric variable or a character variable will do.
   - Even if the response data consists of multiple data items separated with ",", the entire data up to the last data item will be stored in one variable. Be careful when you specify a character variable because, if the array size is not large enough, not all the response data may be stored.
   - Only one input variable can be specified. ";" cannot be specified at the end of the statement.
   - To always send the same inquiry command, use this statement to simplify the program.

Example:

```
GET "CNF?",A$
```

→ Sends a message inquiring for the center frequency and stores the response data in the input parameter A$.

2. COM statement

- This one statement can both send an inquiry command and read the response data. However, only one inquiry command can be written per statement.
- A character constant and/or a variable can be written in an “inquiry command”. The data format for the variable can be specified.
- The response data will be stored in the input variable. If the response data includes a character, specify a character variable. If the response data includes only numerics (numeric characters), either a numeric variable or a character variable will do.
- Multiple input variables can be written. If the response data consists of multiple data items separated with “,”, the data items will be stored in the specified variables one by one.
- “ ; ” cannot be specified at the end of the statement.
- Use this statement if you read the data several times by changing a part of a control message, or send an inquiry message for what is handled as a variable in the program.

Example:

```
COM "MKV? 1, ", I>ML
```

Sends a message inquiring for the marker value at No. I of Trace A and stores the response data in the input variable ML.

Note:

An inquiry message for each marker level in a multi marker configuration will be specified as “MKV? m, n” (m is the trace code, and n is the multi marker number). Therefore, use this statement to read each marker level by changing the n value only.

3. READ 1000 statement

- Reads the response data. This statement will be valid only if an inquiry message is already sent using the PUT or WRITE 1000 statement.
- The response data will be stored in the input variable. If the response data includes a character, specify a character variable. If the response data includes only numerics (numeric characters), either a numeric variable or a character variable will do.
- Multiple input variables can be written. If the response data consists of multiple data items separated with “,”, the data items will be stored in the specified variables one by one.
- To handle the data as one piece even if the response data consists of multiple data items separated with “,”, specify “;” at the end of the statement. The entire data including “,” will be stored in one input variable. In this case, however, only one input variable can be specified. Alternatively, specify only one input variable and execute this statement for multiple times without specifying “;” to read the data separated with “,”.
- If no response data exists, “***” will be output.

Example:

```
WRITE 1ØØØ, "CNF?"
READ 1ØØØ, A$
```

→ Stores in the input parameter A$ the response data for the inquiry command for the center frequency.

4.2.22 Loading and Executing the Program (CHAIN Statement)

(1) Function

Loads and executes the PTA program file on a storage device (internal memory, FD or PMC).

(2) Format

`CHAIN` "File name"

Note:

The command status at the execution of CHAIN (the execution status due to which of the RUN, CONT, or STEP command) will continue even after CHAIN is executed. The break line will also continue to be valid.

4.2.23 ENABLE EVENT Statement

(1) Function

Enables the specified interrupt.
Causes a branch to the event interrupt subroutine specified in the ON EVENT statement when the specified interrupt occurs.

(2) Format

`ENABLE EVENT` I/O number, Factor 3, Factor 2, Factor 1, Factor 0

Notes:

- The I/O number is a numeric variable or a numeric constant.
- The factors 0 to 3 are either a numeric variable/constant, a bit variable/constant or a hexadecimal constant.
- This statement can be directly executed.
- The factors 0 to 3 indicate 32 bits of the I/O interrupt factors.
- The defined bits (b0 to b31) are valid when “1” and invalid when “0”.
- Define the master bit (b31) as “1” to enable all the defined factors regardless of the settings of b0 to b30.

b31 b24	b23 b16	b15 b8	b7 b0
Factor 3	Factor 2	Factor 1	Factor 0

(3) Types of I/O interrupts

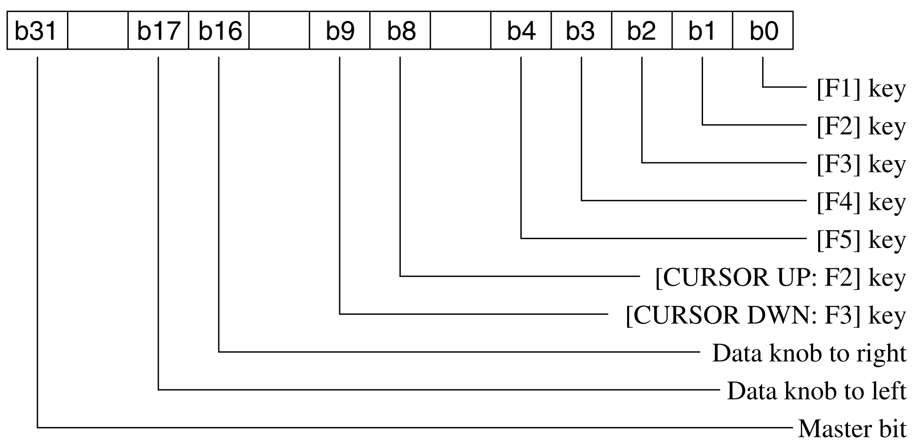
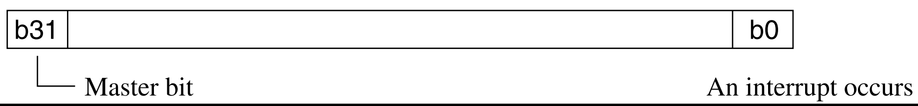
(a) Interrupts from the soft keys and data knob

1. Soft keys [F1] to [F5]
   Pressing the [F1] to [F5] keys (corresponding to system variables EX1 to EX5) in the PTA menu (3/4) will cause an event interrupt. The [F1] to [F5] keys on the PTA keyboard have an equivalent function.
2. Cursor keys
   Pressing the [CURSOR UP: F2] and [CURSOR DWN: F3] keys in the PTA menu (2/4) will cause an event interrupt.
3. Data knob
   Turning the data knob will cause an event interrupt.
   If the operation on the front panel is unlocked, no event interrupt is caused.
   Right and left turns are differentiated.

(b) Interrupts from the parallel I/O port

Receiving a hardware interrupt from the PTA parallel I/O port will cause an event interrupt. You can enable or disable this event regardless of the IOEN, IOMA, and IODI statements.

The following shows the bits corresponding to the I/O interrupt types, I/O numbers, and events.

<table>
<tr><th>I/O type</th><th>I/O No.</th><th>Description</th></tr>
<tr><td>Soft keys and data knob</td><td>11</td><td></td></tr>
<tr><td>PTA parallel I/O port</td><td>41</td><td></td></tr>
</table>

4.2.24 DISABLE EVENT Statement

(1) Function

Disables the specified interrupt.

(2) Format

`DISABLE EVENT` I/O number[, Factor 3, Factor 2, Factor 1, Factor 0]

Notes:

- The I/O number is a numeric variable or a numeric constant.
- The factors 0 to 3 are either a numeric variable/constant, a bit variable/constant or a hexadecimal constant.
- The factors 0 to 3 can be omitted. If omitted, all the interrupt factors will be disabled.
- This statement can be directly executed.
- The defined bits are disabled (invalid) when "1" and the previous valid/invalid status of these bits continues when "0".

4.2.25 ON EVENT Statement

(1) Function

Registers the subroutine to which the program will branch when the specified interrupt occurs.

(2) Format

`ON EVENT` I/O number, Line number (or *Label)

Notes:

- The I/O number is a numeric variable or a numeric constant.
- This statement can be directly executed.
- Use STATUS(M) to identify the interrupt factors. For details, see "Dedicated Functions" in CHAPTER 4.
- If the ON IO GOTO (GOSUB) statement exists and an interrupt from the PTA parallel I/O port occurs, both this statement and the I/O port statement described earlier will be executed. This statement will be executed first.

4.2.26 RETINT Statement

(1) Function

Returns from the event interrupt subroutine.

(2) Format

```
RETINT
```

Notes:

- If another return command is used to return from the event interrupt subroutine, an abort error (F204) will occur.
- If the RETINT command is executed except in an event interrupt, an abort error (F251) will occur.
- You can cause a branch from the event interrupt subroutine to an ordinary subroutine (GOSUB...RETURN).

4.2.27 IOEN Statement

(1) Function

Enables an interrupt from the PTA parallel I/O port.
After the IOEN is executed, the program will branch to the number specified in the ON IO GOTO or ON IO GOSUB statement.

(2) Format

```
IOEN
```

4.2.28 IODI Statement

(1) Function

Disables an interrupt from the PTA parallel I/O port.
After the IODI is executed, the definitions in the ON IO GOTO or ON IO GOSUB statement will be ignored.

(2) Format

```
IODI
```

4.2.29 IOMA Statement

(1) Function

Masks an interrupt from the PTA parallel I/O port.
After IOMA is executed, the definitions in the ON IO GOTO or ON IO GOSUB statement will be ignored. However, if the IOEN statement is executed after an interrupt comes from the I/O port, the program will branch to the previously specified line number.

(2) Format

```
IOMA
```

4.2.30 ON IO GOTO and ON IO GOSUB Statements

(1) Function

Defines the line numbers to which the program should branch when an interrupt from the PTA parallel I/O port occurs.

(2) Format

`ON IO GOTO` Line number or *Label name

`ON IO GOSUB` Line number or *Label name

Note:

The EVENT statement, if existing, will be executed before this statement is executed.

4.2.31 Specifying the Character Size (DCHSIZE Statement)

(1) Function

Specifies the display character size when the system subroutine DCH is executed.

(2) Format

`DCHSIZE` Character size number

Character size number

0 .... Small Font 1 (16 × 8 dots)
1 .... Small Font 2 (16 × 8 dots)
2 .... Large Font (16 × 16 dots)

- The display character size for the PRINT statement, etc. cannot be changed.
- The RESET command, etc. initializes this definition (the initial value is in lowercase 1).
- Lowercase 2 has the same character size as lowercase 1 but its horizontal character intervals are four dots.

4.2.32 Home Position (HOME Statement)

(1) Function

Moves the cursor to its home position (upper left corner).

(2) Format

`HOME`

4.2.33 Erasing (ERASE Statement)

(1) Function

Erases the lines starting from the one where the cursor is located.

(2) Format

```
ERASE
```

Note:

This statement erases the PTA screen only. If a graphic display is used, not all of the screen is erased. In this case, use the system subroutine CALL CFL.

4.2.34 Waiting (WAIT Statement)

(1) Function

Waits for the specified time.

(2) Format

$$\texttt{WAIT}\underbrace{\left\{\begin{array}{l}\text{Numeric constant}\\ \text{Numeric variable}\end{array}\right\}}_{\text{Wait time (in seconds; min. 0.01 seconds)}}$$

4.2.35 Subroutine (CALL Statement)

(1) Function

Executes a system subroutine.
For details of a system subroutine, see “System Subroutines” in CHAPTER 5.

(2) Format

`CALL` System subroutine name [(Parameter [, Parameter ...])]

4.2.36 ON ERROR Statement

(1) Function

Registers a system subroutine to which the program will branch (or which will interrupt) if an error occurs.

(2) Format

`ON ERROR` Line number (or *Label)

Notes:

- If an error occurs while an error handling subroutine is executed, the system will stop executing the subroutine.
- If an error statement is in a line just after the one where an error occurred, only the error statement will be executed.
- If an abort error occurs, no interrupt occurs.
- If an error occurs in the data input for INPUT, no interrupt occurs.
- Use ERRREAD (M) to identify the error code and the line where an error occurred. For details, see “Dedicated Functions” in CHAPTER 4.
- Multiple interrupts together with an event interrupt can occur.
- This statement will not apply to an error that occurred in the error interrupt processing.

4.2.37 OFF ERROR Statement

(1) Function

Cancels the registration of a system subroutine to which the program will branch (or which will interrupt) if an error occurs.

(2) Format

`OFF ERROR`

4.2.38 RETERR Statement

(1) Function

Returns from an error interrupt.
Continues the program from the statement just after the one where an error occurred.

(2) Format

`RETERR`

Notes:

- If the RETURN or RETMAIN command is executed to return from the error interrupt, an abort error (F243) will occur.
- If the RETINT command is executed to return from the error interrupt, an abort error (F251) will occur.
- If the RETERR command is executed when the error interrupt cannot be achieved, an abort error (F252) will occur.
- You can cause a branch from the interrupt routine to an ordinary subroutine (GOSUB···RETURN).

4.2.39 RETRY Statement

(1) Function

Returns from an error interrupt.
Executes the error occurrence statement again to continue the program.

(2) Format

```
RETRY
```

Notes:

- If the RETURN or RETMAIN command is executed to return from the error interrupt, an abort error (F243) will occur.
- If the RETINT command is executed to return from the error interrupt, an abort error (F251) will occur.
- If the RETRY command is executed when the error interrupt cannot be achieved, an abort error (F252) will occur.
- You can cause a branch from the interrupt routine to an ordinary subroutine (GOSUB...RETURN).

4.2.40 RESUME Statement

(1) Function

Returns from an error interrupt.
Continues the program from the specified line.

(2) Format

`RESUME` Line number (or *Label)

Notes:

- If the RETURN or RETMAIN command is executed to return from the error interrupt, an abort error (F243) will occur.
- If the RETINT command is executed to return from the error interrupt, an abort error (F251) will occur.
- If the RESUME command is executed when the error interrupt cannot be achieved, an abort error (F252) will occur.
- You can cause a branch from the interrupt routine to an ordinary subroutine (GOSUB...RETURN).

4.2.41 GIVEUP Statement

(1) Function

Returns from an error interrupt.
Stops executing the program.

(2) Format

`GIVEUP`

Notes:

- If the RETURN or RETMAIN command is executed to return from the error interrupt, an abort error (F243) will occur.
- If the RETINT command is executed to return from the error interrupt, an abort error (F251) will occur.
- If the GIVEUP command is executed when the error interrupt cannot be achieved, an abort error (F252) will occur.
- You can cause a branch from the interrupt routine to an ordinary subroutine (GOSUB...RETURN).

4.2.42 Error Branch (ERROR Statement)

(1) Function

If a recoverable error occurs while a program is executed, use the ERROR statement to continue the execution. The ERROR statement can be written over several lines.
For details, see "ERROR Statement" in CHAPTER 8.

(2) Format

`ERROR` (Error number, Program line to be executed next or *Label name)

4.2.43 Error Main (ERRMAIN Statement)

(1) Function

If a recoverable error (with an error code with W as its initial) occurs while a program is executed, use the ERRMAIN statement to have the program branch to a routine in the topmost layer.

(2) Format

`ERRMAIN` (Error number)

Notes:

- If the ERRMAIN command is executed in the topmost layer, error F213 will occur.
- For details, see "ERRMAIN" in CHAPTER 8.

4.2.44 Data Input 1 (READ Statement)

(1) Function

Receives data from the external device if the external interface port is specified to the controller port.
Reads data from dual port memory if a device port is specified.

(2) Format

`READ` Address, Input variable[, Input variable···]
`READ` Address, Variable[;]

- If " ; " is not added at the end of the statement, the system will regard a comma "," in the received data as the separator of data items, and stores them separately in several input variables.
- If " ; " is added at the end of the statement, the system will not regard a comma "," in the received data as the separator of data items, and stores all the data up to the terminator in one input variable.
- The memory number is specified as an address if reads data from the dual port memory.

4.2.45 Data Input 2 (BREAD Statement)

(1) Function

Receives one byte of binary data from the external device if the external interface port is specified to the controller port.
This statement cannot be executed if a device port is specified.

(2) Format

`BREAD` Address, Input variable[, Input variable···]

4.2.46 Data Input 3 (WREAD Statement)

(1) Function

Receives one word of binary data from the external device if the external interface port is specified to the controller port.
The data will be regarded as upper to lower bytes in the order of sending and stored in input variables.
This statement cannot be executed if a device port is specified.

(2) Format

`WREAD` Address, Input variable[, Input variable···]

4.2.47 Data Output 1 (WRITE Statement)

(1) Function

Sends data to the external device if the external interface port is specified to the controller port.
Writes data to the dual port memory if a device port is specified.

(2) Format

`WRITE` Address, Variable[: Data format][, Variable[: Data format···][;]

- A character constant can also be output.
- If " ; " is added at the end of the statement, the terminator will not be output.
- The memory number is specified as an address if writes data to the dual port memory.

4.2.48 Data Output 2 (BWRITE Statement)

(1) Function

Sends one byte of binary data to the external device if the external interface port is specified to the controller port. This statement cannot be executed if a device port is specified.

(2) Format

`BWRITE` Address, Variable[, Variable...]

Notes:

- Neither the data format or " ; " can be specified.
- The terminator will not be output.

4.2.49 Data Output 3 (WWRITE Statement)

(1) Function

Sends one word (two bytes) of binary data in the order of upper to lower bytes to the external device if the external interface port is specified to the controller port. This statement cannot be executed if a device port is specified.

(2) Format

`WWRITE` Address, Variable[, Variable...]

Notes:

- Neither the data format or " ; " can be specified.
- The terminator will not be output.

Specifing the external interface port

- If one or two digits (e.g., 5 or 17) are specified as an address, the device connected to the port specified in the remote-control PORT command (active port) will be controlled. If three digits (e.g., 105 or 217) are specified as an address, the upper one digit indicates the port number and the lower two digits indicate the address of the device connected to the port specified with the port number. The former will be called the indirect port specification, and the latter the direct port specification.
- As a matter of form, you must specify the address in the indirect port specification and the lower two digits of address in the direct port specification although neither of them are significant for the RS-232C or parallel (Centronics) interface.

Example:

`WRITE␣5,"ABC"` .... Sends data to Address 5 on the specified port (indirect port specification)
`WRITE␣1ØØ,A$` .... Inputs data from a device on Port No.1 (GPIB) (indirect port specification)
`WRITE␣2Ø5,"ABC"` .... Sends data to Address 5 on Port No. 2 (RS-232C) (direct port specification)
`WRITE␣3ØØ,"ABC"` .... Sends data to a device on Port No. 3 (parallel (Centronics)) (direct port specification)

These address specifications are valid for the WRITE, BWRITE, WWRITE, READ, BREAD, WREAD, and LISTG statements.

The following shows the relationship between the port specification commands and the active ports.

	Indirect port specification	Direct port specification		
	WRITE 5	WRITE 1Ø5	WRITE 2Ø5	WRITE 3Ø5
After executing "PORT 1"	GPIB is an active port	GPIB is an active port	RS-232C is an active port (see note)	Parallel (Centronics) is an active port (see note)
After executing "PORT 2"	RS-232C is an active port (see note)	GPIB is an active port	RS-232C is an active port (see note)	Parallel (Centronics) is an active port (see note)
After executing "PORT 3"	Parallel (Centronics) is an active port (see note)	GPIB is an active port	RS-232C is an active port (see note)	Parallel (Centronics) is an active port (see note)

Note:

Specify the address as a matter of form although it is not significant for the RS-232C or parallel (Centronics) interface.

4.2.50 Writing Data to Dual Port Memory (WDPM Statement)

(1) Function

Writes data to a dual port memory.
For details, see “Dual Port Memory” in CHAPTER 6.

(2) Format

`WDPM` Memory number, Variable[: Data format][, Variable[: Data format]...]

Notes:

- A character constant can also be output.
- “ ; ” cannot be specified.
- This statement can be executed regardless of the device/controller status on the external interface port.

4.2.51 Reading Data to Dual Port Memory (RDPM Statement)

(1) Function

Reads data to a dual port memory.
For details, see “Dual Port Memory” in CHAPTER 6.

(2) Format

`RDPM` Memory number, Input variable[, Input variable...]

Notes:

- “ ; ” cannot be specified.
- To enter the data items separated with ", ", specify multiple input variables.
- This statement can be executed regardless of the device/controller status on the external interface port.

4.2.52 SOS (SOS)

(1) Function

Displays SOS in the statement where a syntax error occurred when the program is loaded.

(2) Format

```
Line No.   SOS   Statement of grammatical error
```

Notes:

- A statement with an SOS will be handled as a comment statement as for the REM statement. If this program is executed, it will be handled as a syntax error.
- A line number error will be handled as a syntax error (W6) and no SOS indication is given.

4.2.53 Switching the Write Strobe Signal from the PTA Parallel I/O Port (OLDPORT Statement)

(1) Function

Switches the timing of generating the write strobe pulses that are output when the data is written to Ports C and D in the PTA parallel I/O port.

(2) Format

```
OLDPORT
```

- If this statement is not executed:
  In about 1 µs after the data is written, the write strobe signals of pulse width 1 µs will be output.
- If this statement is executed:
  As soon as the data is written, the write strobe signal of pulses will be terminated (Operation mode B: Compatible to MS6401*)

Notes:

- After the data is written and the stable data is externally output, use "Operation Mode A" to latch and use the signal.
- To detect the timing of data change, use "Operation Mode B".
- At power-on, "Operation Mode A" is initially opted.
- After this statement is executed, Operation Mode B will continue until the next power-on.
- For details of the write strobe signal generation timing, see CHAPTER 9, "CONTROLLING THE PTA PARALLEL I/O PORT".

The timing of generating the write strobe pulses that are output when the data is written to Ports C and D in the PTA parallel I/O port are somewhat different from that of MS3401*. If an error occurs when the write strobe pulses are used in combination with the jigs that used to be connected to the MS3401*, use this statement to change the generation timing to "Compatible to MS3401*". As long as no particular problems are encountered, use of the new generation timing is recommended.

4.2.54 Defining the Pseudo-Random Number (RNDMIZE Statement)

(1) Function

Defines the new initial value for the pseudo-random number sequence generated by the RND function.

(2) Format

```
RNDMIZE
```

Note:

If this statement is not executed, the RND function in the program will generate the same pseudo-random number sequence every time the program is executed.

Section 5 Extended PTL

In addition to PTL which comprises the basic statements, an extended PTL called the system variable and system sub-routine is available. The extended PTL enables calculation and assessment of measuring results as well as control of external devices.

5.1 System Variable

The system variable refers to numeric variables pre-registered in the unit. Therefore, these system variables (variable names) cannot be used as user-defined variables for other purposes. Substituting a value for the system variable chiefly enables setup of measuring conditions and execution of measurements. Moreover, substituting the system variable for the user-defined variable enables loading of measuring conditions and data currently held by the measuring unit to the PTA.

An example where the system variable is used is shown below.

* Setup *
CNF = 10000000
Substituting a numeric constant, 10000000, for the system variable corresponding to the center frequency of “CNF” enables the center frequency of the measuring unit to be set at 10000000 Hz = 10 MHz.

* Loading *
A = CNF
The center frequency value currently registered at the measuring unit is loaded and substituted for the user-defined variable A.

CAUTION ⚠

- **Entering a character row such as “MHz” to substitute for the system variable is not allowed because the system variable only comprises numerical values.**
- **System variables are not available in a way to meet all the measuring conditions. To set up measuring conditions not covered by the pre-defined system variables, use WRITE, 1000 or PUT statement, and execute the normal remote command (GPIB command).**

5.1.1 System Variable - FORMAT/CALIBRATION -

Function	System Variable	Input and Output	Data
FORMAT	TRC	I/O	0 :LOGMAG 1 :PHASE 2 :DELAY 3 :LOGMAG & PHASE 4 :LOGMAG & DELAY 5 :POLAR 6 :IMPD CHART 7 :ADMT CHART 8 :VSWR 9 :LIN MAG 10 :LINMAG & PHASE 11 :LINMAG & DELAY 12 :REAL 13 :IMAGINARY 14 :REAL/IMAGINARY 15 :LOG Z 16 :LOG Z & θ 17 :Q 18 :LOG Z & Q
X → S (Normalization of the sweep.)	CAL	Input Output	1: X → S Starting of the sweep 2: X → S sweep with synchronization 0: X → S Ending of the sweep / 1: During the sweep (Notes)
X → S (Normalization of ON/OFF)	CXS	I/O	0 : OFF/1 : ON
Phase offset	PHO	I/O	–180.000 to +180.000

Notes:

Requesting loading of the sweep status during the sweep process results in a situation where a response data from the measuring unit will not be returned, and the loading request switches into a waiting status. This means that the PTA program and command that follow cannot be executed. The loading request is executed when the sweep ends (sweep suspension status) and the program moves to the next step.

Press the LOCAL key located on the front panel of the measuring instrument to cancel the loading request. The response data will be returned from the measuring unit regardless of whether the unit is in the sweep status.

5.1.2 System Variable - MARKER -

Function	System Variable	Input and Output	Data
Marker function	MKR	Input	0 :NORMAL MARKER 1 :DELTA MARKER 2 :ZERO MARKER 3 :MKR → MAX 4 :MKR → MIN 5 :MKR CHANGE 6 :MKR → CF 7 :DELTA → SPAN 8 :MKR → OFS 9 :MKR → +PEAK 10 :MKR → –PEAK 11 :TRK+PEAK 12 :TRK–PEAK
		Output	0 :NORMAL MARKER 1 :DELTA MARKER 2 :ZERO MARKER
Active market position	CMK	I/O	0 to 1000 (point)
Reference market position	RMK	I/O	0 to 1000 (point)
Marker deletion	MKD	I/O	0 : OFF / 1 : ON
Zero marker value (TRACE-A)	ZRA	Output	
Zero marker value (TRACE-B)	ZRB	Output	
Reference market value (TRACE-A)	RFA	Output	
Reference market value (TRACE-B)	RFB	Output	
Active market value (TRACE-A)	MVA	Output	
Active market value (TRACE-B)	MVB	Output	
Active market frequency	MKF	Output	Output of the active trace

5.1.3 System Variable - DISPLAY -

Function	System Variable	Input and Output	Data
Auto scale	SAU	Input	
Scale (TRACE-A)	SCA	I/O	Same as the remote command (GPIB command) parameter
Scale (TRACE-B)	SCB	I/O	Same as the remote command (GPIB command) parameter
Offset (TRACE-A)	OFA	I/O	Same as the remote command (GPIB command) parameter
Offset (TRACE-B)	OFB	I/O	Same as the remote command (GPIB command) parameter
Sub-trace	STR	I/O	0 :OFF 1 :MT → ST 2 :MT=ST 3 :MT=MT-ST 4 :ON (Input only)
Grid type	DF1	I/O	0 :ALL 1 :CENTER 2 :FRAME
Split display	DF2	I/O	0 : OFF / 1 : ON
Overwrite display	OVP	I/O	0 : OFF / 1 : ON

5.1.4 System Variable - MEASURING CONDITIONS -

Function	System Variable	Input and Output	Data
Center frequency	CNF	I/O	Center frequency numerical value data (Unit: Hz)
Frequency span	SPF	I/O	Frequency span numerical value data (Unit: Hz)
Start frequency	STF	I/O	Start frequency numerical value data (Unit: Hz)
Stop frequency	SOF	I/O	Stop frequency numerical value data (Unit: Hz)
Frequency setup mod	FRQ	I/O	0 : CF-SPAN / 1 : START-STOP
LIN/LOG	LGF	I/O	0 : LIN / 1 : LOG
Sweep time	SWT	I/O	Sweep time numerical value data (Unit: ms)
Automatic setup of the sweep	AU1	I/O	0 : OFF (MANUAL) / 1 : ON (AUTO)
time	AVG	I/O	1 to 1000 (times)
Averaging Number of measuring points	MEP	I/O	0 :11 (points) 1 :21 2 :51 3 :101 4 :251 5 :501 6 :1001
Number of break points	BKP	I/O	1 to 1001 (points)
Total sweep/marker sweep	SW1	I/O	0 : FULL / 1 : MARKER
Resolution bandwidth	RBW	I/O	0 :3 Hz 1 :10 Hz 2 :30 Hz 3 :100 Hz 4 :300 Hz 5 :1 kHz 6 :3 kHz 7 :10 kHz 8 :500 Hz 9 :2 kHz 10 :4 kHz 11 :5 kHz 12 :20 kHz
Automatic setup of the resolution bandwidth	AU2	I/O	0 : OFF (MANUAL) / 1 : ON (AUTO)
Output power/start power	OPL	I/O	Input/output for the output power when the power sweep is OFF and for the start power when the power sweep is ON. Output/start power numerical value data (Unit: dBm)
Power sweep	LSW	I/O	0 : OFF / 1 : ON
Stop power	SOL	I/O	Stop power numerical value data (Unit: dBm)
Step level	SEL	I/O	Step level numerical value data (Unit: dBm)
Input range (TAch)	IRG	I/O	0 : 0 dBm / 1 : +20 dBm
Input impedance (TAch)	IMP	I/O	0 : 50, 75 Ohm / 1 : 1 MOhm
Overload	OVL	Output	0 : NORMAL / 1 : OVER LOAD

5.1.5 System Variable - SWEEP CONTROL -

Function	System Variable	Input and Output	Data
REPEAT sweep and SINGLE sweep	SW2	I/O	0 : REPEAT / 1:SINGLE
Resetting and resumption of the sweep and loading of the sweep status	SW3	Input Output	0 : STOP / 1 : RESET / 2:CONTINUE 0 : Ending of the sweep / 1: During the sweep (Notes)
Starting of the sweep/loading of the sweep status	SWP	Input Output	0 : Starting of the REPEAT sweep/1: Starting of the SINGLE sweep/2: Single sweep (with sweep end synchronization) 0: Ending of the sweep/1: During the sweep (Notes)

Notes:

Requesting loading of the sweep status during the sweep process results in a situation where a response data from the measuring unit will not be returned, and the loading request switches into wait status. This means that the PTA program and command that follow cannot be executed. The loading request is executed when the sweep ends (sweep suspension status) and the program moves to the next step.

Press the LOCAL key located on the front panel of the measuring instrument to cancel the loading request. The response data will be returned from the measuring unit regardless of whether the unit is in the sweep status.

5.1.6 System Variable - SAVE/RECALL -

Function	System Variable	Input and Output	Data
SAVE ITEM parameter	SV1	I/O	0 : OFF / 1 : ON on the active CH
SAVE ITEM S data	SV2	I/O	0 : OFF / 1 : ON on the active CH
SAVE ITEM X data	SV3	I/O	0 : OFF / 1 : ON on the active CH
SAVE ITEM frequency table	SV4	I/O	0 : OFF / 1 : ON on the active CH
SAVE ITEM level table	SV4	I/O	0 : OFF / 1 : ON on the active CH
File delete	DLM	Input	File No. data
Media format	MA4	Inpu	1 : Format execution

5.1.7 System Variable - SYSTEM CONTROL -

Function	System Variable	Input and Output	Data
Title display	TTL	I/O	0 : OFF / /1 : ON
Talker data terminator	TRM	I/O	0 : LF+EOI / 1 : CR&LF+EOI
Time out (when using the controller)	GTM	I/O	Time out data (Unit: sec)
Clock Year	DT0	I/O	0 to 99
Clock Month	DT1	I/O	1 to 12
Clock Day	DT2	I/O	1 to 31
Clock Hour	DT3	I/O	0 to 23
Clock Minute	DT4	I/O	0 to 59
Initialization	INI	Input	1 : Initialization execution

The system variable DT0 is a numeric variable which indicates the lower 2 digits of the Gregorian calendar year. The data range is 0 to 99. When the MS4630B is used to display and print the year data using the following program (as shown below) during the year 2000 to 2009, the data is displayed and printed by one digit (as shown below).

<Program example>

```
100 PRINT DT0
```

<Displayed result at year 2000>

```
0
```

When the year data is required to be printed by 2 digits, create the program as follows.
This program is made to add "0" at the beginning of the year data if the year data of DT0 is between 0 and 9.

<Program example>

```
100 A$=DT0
110 IF DT0=0 LET A$="00"
120 IF DT0=1 LET A$="01"
130 IF DT0=2 LET A$="02"
140 IF DT0=3 LET A$="03"
150 IF DT0=4 LET A$="04"
160 IF DT0=5 LET A$="05"
170 IF DT0=6 LET A$="06"
180 IF DT0=7 LET A$="07"
190 IF DT0=8 LET A$="08"
200 IF DT0=9 LET A$="09"
210 PRINT A$
```

5.1.8 System Variable - WAVEFORM MEMORY, FREQUENCY TABLE DATA, LEVEL TABLE DATA -

The following memory types are mounted on each CH.

Function	System Variable	Input and Output	Data
Waveform memory A (equivalent to TRACE-A)	XMA ( )	I/O	Real type array variable (0 to 1000)
Waveform memory B (equivalent to TRACE-B)	XMB ( )	I/O	Real type array variable (0 to 1000)
S memory A	SMA ( )	I/O	Real type array variable (0 to 1000)
S memory B	SMB ( )	I/O	Real type array variable (0 to 1000)
Image memory A	IMA ( )	I/O	Real type array variable (0 to 1000)
Image memory B	IMB ( )	I/O	Real type array variable (0 to 1000)
Internal measuring memory for the image memory A	IDA ( )	I/O	Floating, real type array variable (0 to 1000)
Internal measuring memory for the image memory B	IDB ( )	I/O	Floating, real type array variable (0 to 1000)
Frequency table	FQM ( )	I/O	Real type array variable (0 to 1000)
Level table	LVM ( )	I/O	Real type array variable (0 to 1000)

5.1.9 Relation between Waveform Memory and Internal Measuring Memory

In addition to the XMA, XMB, SMA, SMB, IMA and IMB waveform memory types, internal measuring memory types corresponding to individual waveform memory types are also mounted on this measuring instrument.

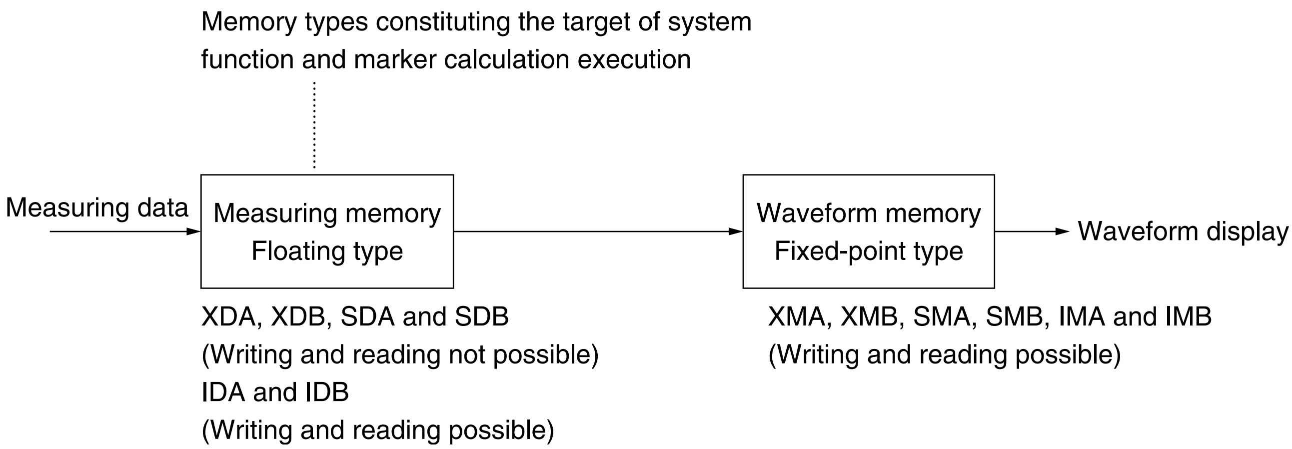
The measuring data obtained from sweeping are first stored into these internal measuring memory types. The data in the measuring memory comes in the form of a floating, real data, which are used to calculate the PTA system function and marker as well as save a storage device (internal memory, FD or PMC).

Reading of and writing into the measuring data memory from the external device are in principle not possible, with exceptions of internal measuring memory types of IDA and IDB which correspond to the image memory types of IMA and IMB respectively.

The waveform memory, on the other hand, stores waveform data constituting the display data, which are generated using the data from the measuring memory. Although reading of and writing into the waveform memory from the external device are allowed, any resultant modifications will not be reflected upon the internal measuring memory.

However, the system sub-routine for memory copy, CALL COPY, enables copying of both waveform memory data and the internal measuring memory data from the source memory to the destination memory.

Moreover, this function also enables writing of any given data into internal memory types of IDA and IDB and copying such data onto other internal measuring memory types using the copy sub-routine. It is also possible to carry out the process from the opposite way, namely copying data from other internal measuring memory types onto IDA and IDB to execute system function.



5.1.10 System Variable - PTA PROGRAM SWITCH/MENU SWITCH -

Function	System Variable	Input and Output	Data
Program switch 1	EX1	I/O	0 : OFF / 1 : ON
Program switch 1	EX2	I/O	0 : OFF / 1 : ON
Program switch 1	EX3	I/O	0 : OFF / 1 : ON
Program switch 1	EX4	I/O	0 : OFF / 1 : ON
Program switch 1 PTA menu switch	EX5	I/O Input	0 : OFF / 1 : ON 2 : Displaying the PTA 1/4 menu. 3 : Displaying the PTA 2/4 menu. 4 : Displaying the PTA 3/4 menu. 5 : Displaying the PTA 4/4 menu.
PTA menu switch	EX6	I/O	0 : Displaying the PTA 1/4 menu. 1 : Displaying the PTA 2/4 menu. 2 : Displaying the PTA 3/4 menu. 3 : Displaying the PTA 4/4 menu.

5.1.11 System Variable - PTA PARALLEL I/O PORT -

Function	System Variable	Input and Output	Data
Output port A	IOA ( )	Output	Output data A (8 bits)
Output port B	IOB ( )	Output	Output data B (8 bits)
I/O port C	IOC ( )	I/O	I/O data C (4 bits)
I/O port D	IOD ( )	I/O	I/O data D (4 bits)
Port C and D I/O switch	EIO	I/O	Port C status / Port D status 0 : Input / Input 1 : Output / Input 2 : Input / Output 3 : Output / Output
External switch (OUTPUT 1, 2)	EX0	I/O	OUTPUT 1 status / OUTPUT 2 status 0 : OFF / OFF 1 : ON / OFF 2 : OFF / ON 3 : ON / ON

5.1.12 System Variable - FILTER FUNCTION I/O -

See the sections of System Function FILTER 1, 2, 3, 4 and 5.

5.2 System Sub-Routine

PTA has exclusive sub-routines called the system sub-routine executed by the CALL statement. The system sub-routines listed below are available.

■ Screen sub-routine

• Display item deletion:	`CALL CER(M)`
• Display item return:	`CALL CRN(M)`
• Screen deletion:	`CALL CFL(M)`
• Character display:	`CALL DCH(X, Y, text, M [, N])`
• Line drawing:	`CALL DLN(XØ, YØ, X1, Y1, M [, N])`
• Square drawing:	`CALL DRC(XØ, YØ, X1, Y1, M [, N])`
• Circle drawing:	`CALL DCR(X, Y, R, M [, N])`
• Arc drawing:	`CALL DAR(XØ, YØ, RØ, W1, W2, M1 [, M3])`
• OX drawing:	`CALL DJG(X, Y, R, F, M1, M2)`
• Soft key label registration:	`CALL DEF(M, text)`

■ File operation sub-routine

• File opening (reading):	`CALL OPNI` character row variable or character constant
• File opening (writing):	`CALL OPNO` character row variable or character constant
• File deletion:	`CALL FDEL` character row variable or character constant
• Data loading:	`CALL DALD` variable
• Data saving:	`CALL DASV` variable
• File closing:	`CALL CLS`

■ GPIB sub-routine (exclusively for the GPIB port)

• Interface clearing: (Switching to the system controller board)	`CALL IFC`
• Service request:	`CALL RSV(M)`
• Switching to the device board:	`CALL DEV`

■ Video plotter sub-routine

• Screen hard copy:	`CALL VPT`

■ Interface sub-routine

• Interface control:	`CALL GPIB` (Port No., control item No.)

■ Panel sub-routine

• Front panel operation LOCK:	`CALL PNLL(Ø)`
• Front panel operation LOCK release:	`CALL PNLU(Ø)`

■ Waveform memory sub-routine

• Memory copy:	`CALL COPY(MØ, M1)`

■ Buzzer sub-routine

- Buzzer calling: `CALL BZR`

CAUTION ⚠

A parameter set outside the predetermined range by the screen sub-routine results in an error, and in this case, it is not possible to draw pictures.

5.2.1 CER, CRN Sub-Routine

(1) Function

This controls the deletion and the display return of the character row, graph, scale and marker.

(2) Style

`CALL␣CER(MØ)` .... Deletion of item M0
`CALL␣CRN(MØ)` .... Display return of item M0

M0	Item
0	All items
1	Trace A line term (CH 1, 2)
2	Trace B line term (CH 1, 2)
3	Measuring condition term (CH 1, 2)
4	Frequency term (CH 1, 2)
5	Menu term
6	Sweep marker term (CH 1, 2)
7	Scale A (CH 1, 2)
8	Scale B (CH 1, 2)
9	Waveform A (CH 1, 2)
10	Waveform B (CH 1, 2)
11	Marker A (CH 1, 2)
12	Marker B (CH 1, 2)
13	Marker frequency and clock term (CH 1, 2)

Note:

- M0 represents the numerical value constant or numerical value variable.
- When data is erased or display-returned by this subroutine, the current situation retains until the setting or PTA determined by the subroutine is turned OFF.
- The marker list, Filter analysis result and Resonator analysis result are deleted or restored when M 0 = 0 (all items) is specified.

5.2.2 CFL Sub-Routine

(1) Function

This deletes all the screen range which corresponds to the specified subtraction.

(2) Style

```
CALL CFL(M1)
```

M1 (Screen designation)	Screen No. constituting the deletion target
0	All screens
1	7th screen (PTA screen)
2	1st screen (Error message screen)
3	2nd screen (Directory screen)
4	3rd screen (Character screen within the window)
5	4th screen (Background screen within the window)
6	5th screen (Marker screen)
7	6th screen (Marker screen)
8	7th screen (PTA screen)
9	8th screen (Waveform and condition screen)
10	9th screen (Waveform and condition screen)
11	10th screen (Limit line or Waveform screen)
12	11th screen (Limit line or Waveform screen)
13	12th screen (Storage waveform screen)
14	13th screen (Scale screen)
15	14th screen (Scale screen)
16	15th screen (marker frequency and menu character screen)
17	16th screen (Message screen)

Notes:

- M1 represents the numerical value constant or numerical value variable.
- This sub-routine temporarily deletes the screen. Therefore, when another set of display conditions are established (for example, the measuring parameter value has been modified, or characters or graphics are displayed), these new conditions are displayed.
- Each screen has preset display priority. When a figure is drawn on the same coordinate of multiple screens, it is displayed by the attribute (color) of the screen which takes the first priority among all the screens used. The screen No. 1 is prioritized first, while the screen No. 16 is prioritized last.

5.2.3 DCH Sub-Routine

(1) Function

This displays the character row. (The bottom left of the screen is set as reference.)

(2) Style

```
CALL␣DCH(X,Y,text,M1[,M2])
```

Normal display/reversed display
Screen designation
Character row displayed
Y coordinate of the header character
X coordinate of the header character

M2	Line type
0	Normal display
1	Reversed display

Range of each parameter

	Header X coordinate (X)	Header Y coordinate (Y)	Maximum character row (text)
Small font 1	0 to 632	0 to 464	80
Small font 2	0 to 628	0 to 464	53
Large font 3	0 to 624	0 to 464	40

Notes:

- The header X and Y coordinates refer to the bottom left corner position of a character.
- X, Y, M1 and M2 represent the numerical value constant or numerical value variable. text represents the character constant or character row variable.
- M2 can be omitted. However, the default set for M2, when it is omitted, is 0.
- DCHSIZE statement allows setup of the character size (small font/large font).
  DCHSIZE 0: Small font 1
  DCHSIZE 1: Small font 2
  DCHSIZE 2: Large font
- The following remote control command (GPIB command) can move the origin of the drawing coordinate.
  "GORG_x, y" : Moves the origin of the drawing coordinate from the left bottom corner (0, 0) to (x, y).
  $-639 \le x \le 639$
  $-479 \le y \le 479$

5.2.4 DLN Sub-Routine

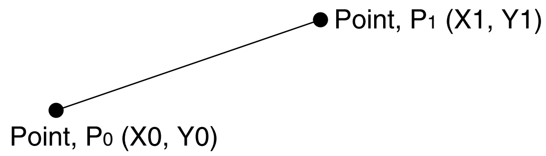
(1) Function

This displays the straight line (stroke).

(2) Style

```
CALL␣DLN(XØ,YØ,X1,Y1,M1[,M3])
```

- M3: Line type designation
- M1: Screen designation
- Y1: Y coordinate of point, P1
- X1: X coordinate of point, P1
- YØ: Y coordinate of point, P0
- XØ: X coordinate of point, P0



M3	Line type
0	Displayed by an unbroken line
1	Deleted by an unbroken line
2	Displayed by a broken line
3	Deleted by a broken line

Notes:

- X0, Y0, X1, Y1, M1 and M3 represent the numerical value constant or numerical value variable.
- M3 can be omitted. However, the default set for M3, when it is omitted, is 0.
- The following remote control command (GPIB command) can move the origin of the drawing coordinate.

  "GORG_x, y" : Moves the origin of the drawing coordinate from the left bottom corner (0, 0) to (x, y).

  $-639 \le x \le 639$

  $-479 \le y \le 479$

5.2.5 DRC Sub-Routine

(1) Function

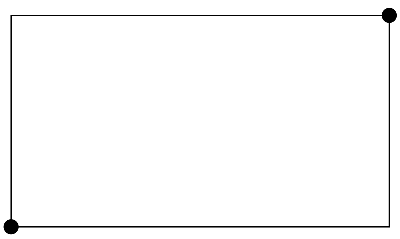
This displays the square with two specified points set diagonally.

(2) Style

```
CALL␣DRC(XØ,YØ,X1,Y1,M1[,M3])
```

Line type designation
Screen designation
Y coordinate of point, P1
X coordinate of point, P1
Y coordinate of point, P0
X coordinate of point, P0

Point, P_1 (X1, Y1)



Point, P_0 (X0, Y0)

M3	Line type
0	Displayed by an unbroken line
1	Deleted by an unbroken line
2	Displayed by a broken line
3	Deleted by a broken line

Notes:

- X0, Y0, X1, Y1, M1 and M3 represent the numerical value constant or numerical value variable.
- M3 can be omitted. However, the default set for M3, when it is omitted, is 0.
- When P0 (X0, Y0) and P1 (X1, Y1) are on the same coordinate, no square is displayed.
- The following remote control command (GPIB command) can move the origin of the drawing coordinate.

  "GORG_x, y" : Moves the origin of the drawing coordinate from the left bottom corner (0, 0) to (x, y).

  $-639 \leq x \leq 639$

  $-479 \leq y \leq 479$

5.2.6 DCR Sub-Routine

(1) Function

This displays the circle.

(2) Style

```
CALL␣DCR(X,Y,R,M1[,M3])
```

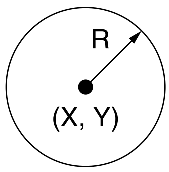
Line type designation

Screen designation

Radius (0 <)

Y coordinate of the center

X coordinate of the center



M3	Line type
0	Displayed by an unbroken line
1	Deleted by an unbroken line
2	Displayed by a broken line
3	Deleted by a broken line

Notes:

- X, Y, R, M1 and M3 represent the numerical value constant or numerical value variable.
- M3 can be omitted. However, the default set for M3, when it is omitted, is 0.
- The following remote control command (GPIB command) can move the origin of the drawing coordinate.

  "GORG_x, y" : Moves the origin of the drawing coordinate from the left bottom corner (0, 0) to (x, y).

  $-639 \le x \le 639$

  $-479 \le y \le 479$

5.2.7 DAR Sub-Routine

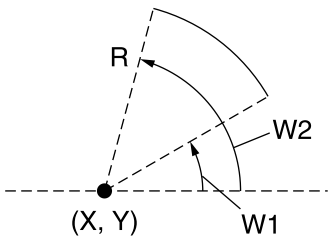
(1) Function

This displays the arc.

(2) Style

```
CALL␣DAR(X,Y,R,W1,W2,M1[,M3])
```

Line type designation
Screen designation
Angle at which to end drawing (–180.00 to 180.00)
Angle at which to start drawing (–180.00 to 180.00)
Radius (0 <)
Y coordinate of the center
X coordinate of the center



M3	Line type
0	Displayed by an unbroken line
1	Deleted by an unbroken line
2	Displayed by a broken line
3	Deleted by a broken line

Notes:

- X, Y, R, W1, W2, M1 and M3 represent the numerical value constant or numerical value variable.
- M3 can be omitted. However, the default set for M3, when it is omitted, is 0.
- The following remote control command (GPIB command) can move the origin of the drawing coordinate.

  "GORG_x, y" : Moves the origin of the drawing coordinate from the left bottom corner (0, 0) to (x, y).

  $-639 \le x \le 639$

  $-479 \le y \le 479$

5.2.8 DEF Sub-Routine

(1) Function

This registers the menu label (name) at the soft key menu. When the PTA menu (3/4) is displayed, the label registered by this sub-routine is indicated.

(2) Style

```
CALL␣DEF(M1,text)
```

Label comprising a maximum of 10 characters

Soft key Nos. (1 to 6)

Notes:

- M represents the numerical value constant or numerical value variable.
- "text" represents the character constant or character row variable.
- The label registered by this sub-routine is effective until PTA is turned OFF.

5.2.9 OPNI, OPNO and FDEL Sub-Routine

(1) Function

This opens the data file to execute data writing in and reading from a storage device (internal memory, FD or PMC), as well as deletes the data file created.

(2) Style

`CALL␣OPNI` Character row variable or character constant : File opening to read data

`CALL␣OPNO` Character row variable or character constant : File opening to write data

`CALL␣FDEL` Character row variable or character constant : Data file deletion

Notes:

- The data file name, which begins with a %, constitutes a row of up to six alphanumeric characters including the %.
- Do not eject the FD or PMC while the data file is open.
- This sub-routine cannot be applied to the PTA program file located.

5.2.10 DALD and DASV Sub-Routine

(1) Function

This loads (DALD) the data stored on the storage device (internal memory, FD or PMC), or saves (DASV) the data.

(2) Style

`CALL␣DALD␣` input variable: Data loading from the data file

`CALL␣DASV␣` variable: Data saving to the data file

Notes:

- The data file is created as the sequential file. Therefore, load the data in the same order as that of data saving.
- Saving different data types (for example, numerical value type data and character type data) in a single data file is also possible. However, when the data type used for saving and the input variable type used for loading cannot substitute for each other, saving such data results in an error.

(3) Program example

```
1Ø  REM "*** DATA FILE ***"
2Ø  CALL OPNO"%DATA"
3Ø  FOR C=2 TO 1Ø
4Ø  D=C * C
5Ø  CALL DASV D
6Ø  NEXT C
7Ø  CALL CLS
8Ø  CALL OPNI"%DATA"
9Ø  FOR C=2 TO 1Ø
1ØØ CALL DALD D
11Ø PRINT "RES.=",D
12Ø NEXT C
13Ø CALL CLS
14Ø STOP
```

Lines 1Ø–6Ø: Saving to the file

Lines 8Ø–13Ø: Loading from the file

Execution results

```
RES.=4
RES.=9
RES.=16
RES.=25
RES.=36
RES.=49
RES.=64
RES.=81
RES.=1ØØ
```

5.2.11 CLS Sub-Routine

(1) Function

This closes the data file opened. This function is shared for data saving and loading.

(2) Style

`CALL␣CLS`

5.2.12 IFC Sub-Routine

(1) Function

The GPIB port, functioning as the system controller board, outputs the interface clearing signal to the device connected on the GPIB bus.

(2) Style

`CALL␣IFC`

5.2.13 RSV Sub-Routine

(1) Function

The GPIB port, when used as the device port, sends the service request to the controller.

(2) Style

`CALL␣RSV (M)`

M	PTA Event Status Register MSB							LSB
0	×	×	×	×	0	0	0	1
1	×	×	×	×	0	0	1	0
2	×	×	×	×	0	0	1	1
3	×	×	×	×	0	1	0	0
4	×	×	×	×	0	1	0	1
5	×	×	×	×	0	1	1	0
6	×	×	×	×	0	1	1	1
7	×	×	×	×	1	0	0	0
8	×	×	×	×	1	0	0	1
9	×	×	×	×	1	0	1	0

The × refers to a bit that remains unchanged.

The PTA event status register is defined as the bit 1 extended status of the status byte. Therefore, when the data shown on the left are set for the PTA event status register, bit 1 of the status byte is indirectly set as the summary bit. Moreover, as the logical sum of all the bits of the status byte, the RQS bit (bit 6) is set to send the service request to the controller. The GPIB commands used to load the status byte and the PTA event status register from the external controller are *STB? and ESRI? respectively.

Notes:

- M represents the numerical value constant or numerical value variable.
- Only effective when the GPIB port functions as the device port.

5.2.14 DEV Sub-Routine

(1) Function

Executing this sub-routine when the GPIB port is functioning as the system controller port switches the GPIB port to the device port.

(2) Style

`CALL␣DEV`

5.2.15 Interface Control Sub-Routine (GPIB)

(1) Function

This enables the designated port to perform the following tasks; sending of "Interface Clear" (IFC), sending of "Remote" (REN), setup of "Return to Local" (RTL), sending of "Local" (LCL), sending of "Device Clear" (DCL), sending of "Local Lockout" (LLO) and sending of "Device Trigger" (DTR).

(2) Style

`CALL␣GPIB (1, Ø)`	Sending of IFC
`CALL␣GPIB (1, 1[, Address])`	Setup of REN
`CALL␣GPIB (1, 2)`	Sending of RTL
`CALL␣GPIB (1, 3[, Address])`	Sending of LCL
`CALL␣GPIB (1, 4[, Address])`	Sending of DCL
`CALL␣GPIB (1, 5)`	Sending of LLO
`CALL␣GPIB (1, 6, Address)`	Sending of DTR

Address: 0 to 30 device GPIB address

Notes:

- Address represent numerical value constant or numerical value variable.
- Operations in each sub-routine are as follows.

IFC:

- This turns On the IFC line and initializes the interface function of all the devices connected.
- This initializes only the interface function, without affecting other device functions.
- No talkers or listeners are canceled.
- This does not affect the SRQ line.

REN:

- Omitting [, Address] turns On the REN line. Following this, when individual devices are designated as the listener, they switch into the remote control status.
- Specifying [, Address] turns On the REN line and designates the devices specified by [, Address} as the listener, switching them into the remote control status.
- This task can be executed only when the GPIB port is functioning as the system controller port.

RTL:

- The GPIB port switches the measuring instrument to the local control status.
  (Equivalent to the status where the LOCAL key is pressed.)

LCL:

- Omitting [, Address] turns OFF the REN line, switching all the devices into the local control status.
- Specifying [, Address] cancels all the listeners, and then, set all the devices specified by [, Address] as the listener, switching them into the local control status. The REN line remains unchanged.
- This task can be executed only when the GPIB port is functioning as the system controller port.

DCL:

- Omitting [, Address] prompts sending of “DCL” and initializes the function of all the devices on GPIB.
- Specifying [, Address] prompts sending of “SDC” (Selected Device Clear) and initializes the function of devices specified by [, Address].
- This task can be executed only when the GPIB port is functioning as the system controller port.

LLO:

- This sets the switching from remote to local of all the devices on GPIB as ineffective. This disables the LOCAL key on the panel to switch devices on GPIB into the local control status
- Switching from PTA using the REN and LCL instructions is possible.
- This status is canceled by the LCL instructions with its [, Address] specification omitted.
- This task is effective only when the GPIB port is functioning as the system controller port.

DTR:

- This sends out the specified device trigger. The specified devices commence preset operations.
- This task is effective only when the specified port is functioning as the system controller port.

5.2.16 PNLU and PNLL Sub-Routine

(1) Function

This sets up LOCK/UNLOCK of the front panel when PTA is ON.

(2) Style

`CALL␣PNLU(Ø)` Front panel UNLOCK
`CALL␣PNLL(Ø)` Front panel LOCK

Note:

Switching the soft keys F1 - F6, SHIFT key, ten key and LOCAL key on the front panel into the LOCK status is not possible.

5.2.17 COPY Sub-Routine

(1) Function

This copies the waveform memory (copy source) data specified onto other waveform memory types (copy destination).

(2) Style

```
CALL␣COPY(MØ,M1)
```

Waveform memory No. serving as the copying destination

Waveform memory No. serving as the copying source

M0, M1	Waveform memory	System variable name	Type
0	CH1 waveform memory A	XMA ( )	Real number
1	CH1 waveform memory B	XMB ( )	Real number
2	CH1 S memory A	SMA ( )	Real number
3	CH1 S memory B	SMB ( )	Real number
4	CH1 image memory A	IMA ( )	Real number
5	CH1 image memory B	IMB ( )	Real number
6	(Unused)		
7	(Unused)		
8	CH2 waveform memory A	XMA ( )	Real number
9	CH2 waveform memory B	XMB ( )	Real number
10	CH2 S memory A	SMA ( )	Real number
11	CH2 S memory B	SMB ( )	Real number
12	CH2 image memory A	IMA ( )	Real number
13	CH2 image memory B	IMB ( )	Real number

Note:

- The contents of M0 are copied onto M1. The M0 contents remain unchanged. The older M1 contents will be lost.
- M0 and M1 uses the numerical constant or variable.
- See System Variable under Chapter 5 for the details of the memory copy.

5.2.18 DJG Sub-Routine

(1) Function

This subroutine draws ○ or × in double lines at the specified coordinates.
Alternately, it deletes × to display ○, or deletes ○ to display ×.

(2) Format

CALL DJG (X, Y, R, F, M1, M2)

- M2: Screen number at which to display or delete ×
- M1: Screen number at which to display or delete ○
- F: Drawing function
- R: Radius of circle (larger or equal to 1)
- Y: Y-coordinate of center
- X: X-coordinate of center

Drawing function (F)	Description
0	Clears an existing × and displaying ○ at the same coordinates
1	Clears an existing ○ and displaying × at the same coordinates

Notes:

- For X, Y, R, F, M1, and M3, use a numerical constant or variable.
- If M1 is set to -1, ○ is neither displayed or deleted.
- If M2 is set to -1, × is neither displayed or deleted.
- The origin of drawing coordinates may be moved.

5.2.19 BZR Sub-Routine

(1) Function

This subroutine calls the buzzer once.

(2) Format

CALL BZR

5.3 System Function

The system function enables high-speed extraction and calculation of feature points, targeting internal measuring memory (memory where the measuring data serving as the source for the waveform memory data are stored). Therefore, a value exists as the function calculation result.

System Function		Function
Maximum value	MAX (M, P0, P1, N)	Used to calculate the maximum value between P0 and OP1.
Minimum value	MIN (M, P0, P1, N)	Used to calculate the minimum value between P0 and OP1.
Frequency of the specified measuring value 1	BNDL (M, P0, L, N)	Used to begin search starting from P0 and calculate the frequency of the specified measuring value.
Frequency of the specified measuring value 2	BNDH (M, P0, L, N)	Used to begin search starting from P0 and calculate the frequency of the specified measuring value.
Frequency of the specified measuring value 3	MEAS (M, P0, L, N)	Used to begin search starting from P0 and calculate the frequency of the specified measuring value.
Ripple 1	RPL1 (P0, P1, N[, R])	Used to calculate the ripple between P0 and P1.
Ripple 2	RPL2 (P0, P1, N[, R])	Used to calculate the ripple between P0 and P1.
Ripple 3	RPL3 (P0, P1, N[, R])	Used to calculate the ripple between P0 and P1.
Ripple 4	RPL4 (P0, P1, N[, R])	Used to calculate the ripple between P0 and P1.
Pole 1	POLL (M, P0, L, N[, R])	Used to begin search starting from P0 and calculate the pole (dip).
Pole 2	POLH (M, P0, L, N[, R])	Used to begin search starting from P0 and calculate the pole (dip).
Extreme-high value	PLRH (M, P0, N[, R])	Used to begin search starting from P0 and calculate the adjacent extreme-high value.
Extreme-low value	PLRL (M, P0, N[, R])	Used to begin search starting from P0 and calculate the adjacent extreme-low value.
Filter 1	FILTER1 (N)	Used to calculate the insertion loss, bandwidth and ripple.
Filter 2	FILTER2 (N)	Used to calculate the maximum delay, the difference between the maximum and minimum delay and center frequency delay.
Filter 3	FILTER3 (N)	Used to calculate the attenuation bandwidth and attenuation.
Filter 4	FILTER4 (N)	Used to calculate the attenuation bandwidth and attenuation.
Filter 5	FILTER5 (N)	Used to calculate the minimum attenuation value.
Group delay	GPDLY (P0, P1, CH[, E])	Used to calculate the group delay time at the specified measuring point.
Status byte loading	GST (G)	Reads the status byte from the external equipment of the specified address.

CAUTION ⚠

- **As the system function targets the internal measuring memory, the value input (P0 and P1) to each function must be specified as the point value on the memory.**
- **P0, P1, L, N and R, representing the numerical value constant or numerical value variable, are input parameters.**
- **M, which represents the numerical value variable, is an output parameter.**
- **N, N1 and N2, which constitute the parameters to specify the waveform memory, represent the numerical value constant or numerical value variable.**
- **G specifies the GPIB address of the device.**

CAUTION ⚠

The internal measuring memory constitutes the target of actual execution in the system function calculation to specify the waveform memory. The both data generally comply with each other under normal measurements. (When the measuring format is LOGZ, the linear data and the logarithm data are stored on the internal measuring memory and the waveform memory respectively.) However, when the data on the waveform memory are written over by system functions (such as XMA, XMB, SMA and SMB), the modified contents are not reflected upon the internal measuring memory, causing a discrepancy between the data on the waveform memory and the data on the internal measuring memory.
Even under this situation, the internal measuring memory constitutes the target of actual execution of the system function calculation. See the section of System Variable under Chapter 5 for more details of the waveform memory and the internal measuring memory. Under this chapter, the following section provides explanations on the assumption that the data on the waveform memory and the data on the internal measuring memory comply with each other.

N,N1,N2	Waveform memory A	Reference setup format and resolution
0	CH1 waveform memory A	Measuring format and resolution of CH1 TRACE-A
1	CH1 waveform memory B	Measuring format and resolution of CH1 TRACE-B
2	CH1 S memory A	Measuring format and resolution of CH1 TRACE-A
3	CH1 S memory B	Measuring format and resolution of CH1 TRACE-B
4	CH1 image memory A	Measuring format and resolution of CH1 TRACE-A
5	CH1 image memory B	Measuring format and resolution of CH1 TRACE-B
6	(Unused)	
7	(Unused)	
8	CH2 waveform memory A	Measuring format and resolution of CH2 TRACE-A
9	CH2 waveform memory B	Measuring format and resolution of CH2 TRACE-B
10	CH2 S memory A	Measuring format and resolution of CH2 TRACE-A
11	CH2 S memory B	Measuring format and resolution of CH2 TRACE-B
12	CH2 image memory A	Measuring format and resolution of CH2 TRACE-A
13	CH2 image memory B	Measuring format and resolution of CH2 TRACE-B

Notes:

The output of the measuring value as a result of function calculation and the output of calculation of the measuring value use, as reference, the measuring format and resolution shown above which correspond to the relevant waveform memory. The data output is the fixed-point real type, like the marker value. Moreover, the specified measuring value and specified resolution (L, R) are interpreted as the unit in compliance with the above measuring format and resolution.

CAUTION ⚠

When the measuring format is LOGZ, the LINZ (linear value) is stored on the internal measuring memory. Therefore, a large expected value of the function calculation result may result in an overflow when it is converted into the fixed-point real type data.

5.3.1 MAX Function

(1) Function

This calculates the maximum value and the measuring point within the waveform memory range specified.

(2) Style

```
MAX(M,PØ,P1,N)
 |  |   |   |
 |  |   |   Waveform memory designation
 |  |   Range designation (P0<P1)
 |  Measuring point of the maximum value
 Maximum value
```

Note:

When the same maximum values exist in multiple numbers, the first point of the maximum value is stored on M.

(3) Program example: Calculation of the maximum level of the TRACE-A on the waveform memory CH1.

```
1Ø REM"MAX (M, PØ, P1, N)"
2Ø GMAX=MAX (M, Ø, 5ØØ, Ø)
3Ø'
4Ø PRINT "Maximum Level=",GMAX,"dB"
5Ø STOP
Maximum Level=–2Ø.45dB
```

5.3.2 MIN Function

(1) Function

This calculates the minimum value and the measuring point within the waveform memory range specified.

(2) Style

```
MIN(M,PØ,P1,N)
 |  |   |   |
 |  |   |   Waveform memory designation
 |  |   Range designation (P0<P1)
 |  Measuring point of the minimum value
 Minimum value
```

Note:

When the same minimum values exist in multiple numbers, the first point of the minimum value is stored on M.

(3) Program example: Calculation of the minimum level of the TRACE-A on the waveform memory CH1.

```
1Ø GMIN=MIN (M, Ø, 5ØØ, Ø)
2Ø'
3Ø PRINT"Min Level=", GMIN,"dB at", M
4Ø STOP
```

5.3.3 RPL1 and RPL2 Function

(1) Function

This calculates ripple 1 and ripple 2 within the specified range of the specified waveform memory.

Ripple 1: Calculates the difference between the largest local maximum and the smallest minimum values.

Ripple 2: Calculates the maximum value of difference between the neighboring local maximum and minimum.

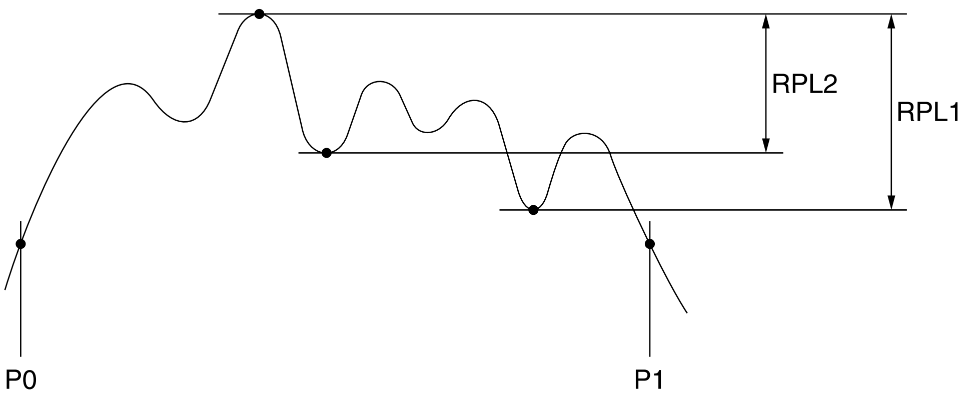
(2) Style

```
RPL1(PØ,P1,N[,R])
RPL2(PØ,P1,N[,R])
```

R: Ripple resolution

N: Waveform memory designation

PØ,P1: Range designation (P0<P1)



Note:

When the difference between the adjacent extreme-high and extreme-low levels is smaller than R, this is not considered a ripple.

(3) Program example

Calculate ripple 1 between the measuring points of 100 and 300 on CH1 TRACE-A. However, set the resolution at 0.2 dB in this case.

```
1Ø RP=RPL1 (1ØØ, 3ØØ, Ø, Ø.2)
2Ø PRINT“RPL1”, RP,“dB”
3Ø STOP
```

5.3.4 RPL3 Function

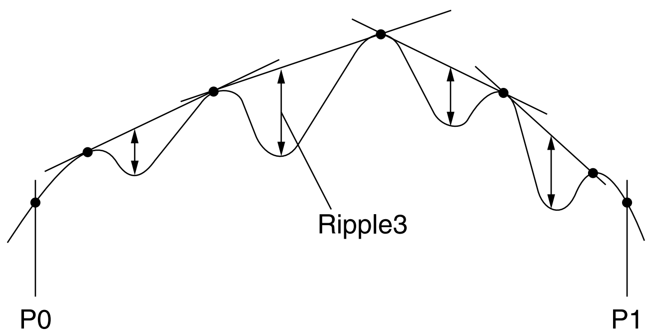
(1) Function

This function calculates, as ripple 3, the maximum value of difference between the straight line connecting two neighboring local maximum points and the local minimum point located between them, in the setup range of specified waveform memory as shown by the figure below.

(2) Style

```
RPL3(PØ,P1,N[,R])
```

Ripple resolution
Waveform memory designation
Range designation (P0<P1)



Note:

When the difference between the adjacent extreme-high and extreme-low levels is smaller than R, this is not considered a ripple.

(3) Program example

Calculate ripple 3 between the measuring points of 50 and 450 on CH1 TRACE-A. However, set the resolution at 0.1 dB in this case.

```
1Ø RP=RPL3 (5Ø, 45Ø, Ø, Ø.1)
2Ø PRINT"RPL3=", RP,"dB"
3Ø STOP
```

5.3.5 RPL4 Function

(1) Function

This designates, as ripple 4, the difference between the waveform memory data and what corresponds to the maximum extreme-high point and the minimum extreme-low point of the absolute value data calculated by differentiating the waveform memory data in the frequency axial in the preset range of the specified waveform memory.

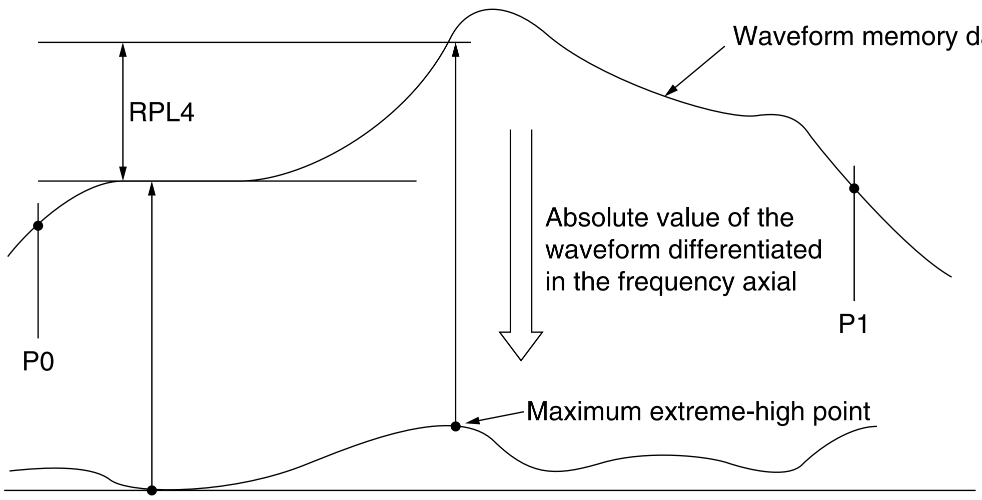
(2) Style

```
RPL4(PØ,P1,N[,L])
```

Ripple resolution

Waveform memory designation

Range designation (P0+3 ≤ P1)



Note:

When the difference between the differentiated extreme-high and extreme-low points is smaller than L, this is not considered a ripple.

Calculation procedures:

1. Calculate all the data on the difference to the adjacent points concerning individual measuring data of the waveform memory (X).
2. Calculate the absolute values for individual data calculated in 1, and set data based on the calculation results as waveform Y.
3. Calculate the maximum extreme-high point (generally, referring to the point where the waveform memory (X) experiences the steepest tilt) and the minimum extreme-low point (generally, referring to the point where the waveform memory (X) experiences the least steep tilt) of the waveform Y.
4. Calculate the waveform memory (X) data of Ymax and Ymin, which correspond to the extreme-high point and extreme-low point calculated in 3 respectively.
5. Designated, as ripple 4 (function result), the Ymax–Ymin absolute value.

5.3.6 BNDL, BNDH and MEAS Function

(1) Function

This calculates the frequency of the specified measuring value with a point set as the starting point, in the specified waveform memory. When no measuring point which corresponds to the specified measuring value exists, the frequency is calculated by directly interpolating from the measuring points located before and behind.

(2) Style

```
BNDL(M,PØ,L,N)
BNDH(M,PØ,L,N)
MEAS(M,PØ,L,N)
```

N: Waveform memory designation
L: Specified measuring value
PØ: Point of search start
M: Measuring point of the specified measuring value
BNDL/BNDH/MEAS: Frequency (or time) of the specified measuring value

When specifying N = 0 to 5:

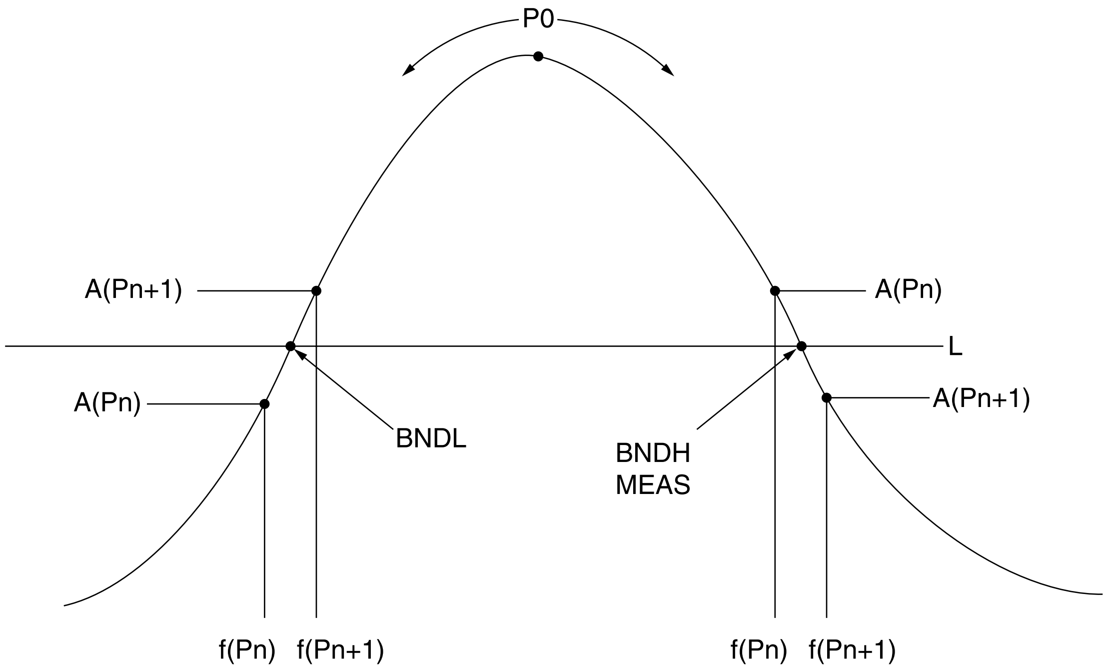
The frequency of the specified measuring value is calculated from the CH1 preset frequency.

When specifying N = 8 to 13:

The frequency of the specified measuring value is calculated from the CH2 preset frequency.

Note:

BNDH and MEAS functions fulfill the same performance.



$$\left\{\begin{matrix} \text{BNDL} \\ \text{BNDH} \\ \text{MEAS} \end{matrix}\right\} = f\,(Pn) + \frac{f\,(Pn+1) - f\,(Pn)}{A\,(Pn) - A\,(Pn+1)} \times \{A\,(Pn) - L\}$$

Note:

When no specified measuring value exists, M equals zero for BLDL, while M equals 1,000 for BNDH and MEAS.

(3) Program example:

Calculate the bandwidth at –20 dB level of the CH1 TRACE-A. (searched from the center position)

```
1Ø L=-2Ø ............................................................ indicating –20 dB.
2Ø FH=BNDL (ML, 25Ø, L, Ø)
3Ø FH=BNDH (MH, 25Ø, L, Ø)
4Ø BW=(FH-FL)/1ØØØ
5Ø PRINT“BW=", BW, "KHz"
6Ø STOP
```

5.3.7 PLRH and PLRL Function

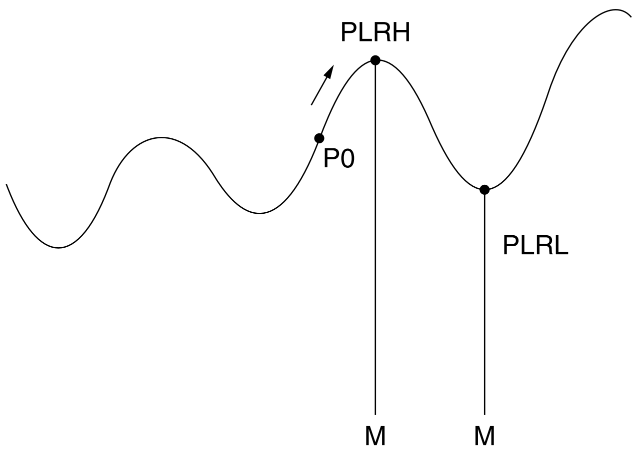
(1) Function

This calculates the first extreme-high and extreme-low values and the points where they are measured, with a point set as the starting point, in the specified waveform memory.

(2) Style

```
PLRH(M,PØ,N[,R]):extreme-high point
PLRL(M,PØ,N[,R]):extreme-low point
```

Ripple resolution
Measuring memory designation
Point of search start
Measuring points of the extreme-high and extreme-low values
Extreme-high and extreme-low values



Notes:

- When the difference between the adjacent extreme-high and extreme-low levels is smaller than R, it is not considered a point of inflection. When R is omitted, R is set at a default of zero.
- When no extreme-high or extreme-low point exists, M equals 1,000, and the value measured at each point becomes the value for the point.

(3) Program example

Calculate the extreme-high value located to the right of CH1 TRACE-A point of 250. However, set the resolution at 3 dB in this case.

```
1Ø PL=PLRH (M, 25Ø, Ø, 3)
2Ø PRINT“Peak Level=", PL, "dB at", M
3Ø STOP
```

5.3.8 POLL and POLH Function

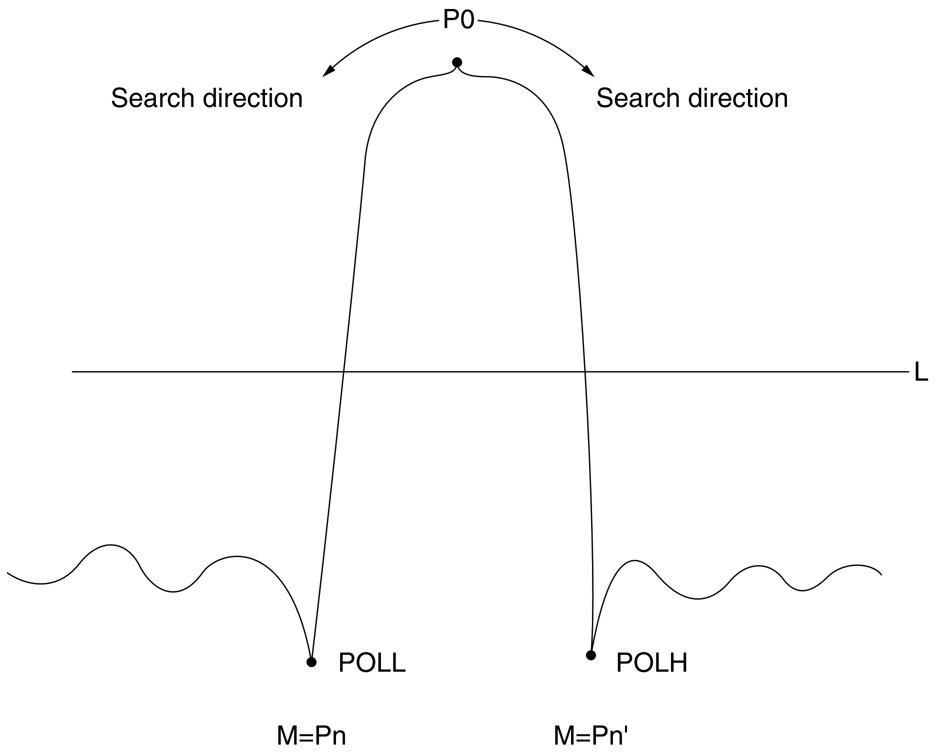
(1) Function

As shown by the figure below, this calculates the pole located in the measuring range which comes below the specified measuring value with a point set as the staring point, in the specified waveform memory.

(2) Style

```
POLL(M,PØ,L,N[,R]):Search in the left direction
POLH(M,PØ,L,N[,R]):Search in the right direction
```

Ripple resolution
Waveform memory designation
Specified measuring point
Point of search start
Measuring point of the pole



Notes:

- A failure to detect a pole in the POLL function results in M equaling zero, and the value measured at point zero is set as POLL.
- A failure to detect a pole in the POLH function results in M equaling 1,000, and the value measured at this point is set as POLH.
- When the difference between the adjacent extreme-high and extreme-low levels is smaller than R, the extreme point level in this case is not considered a pole.

(3) Program example

Search and calculate the pole level smaller than -60 dB in the range located to the left of point, 250, within CH1 TRACE-A. However, set the resolution at 1 dB in this case.

```
1Ø PL=POLL (M, 25Ø, -6Ø, Ø, 1)
2Ø PRINT"Poll Level=", PL, "dB at", M
3Ø STOP
```

5.3.9 FILTER 1 Function

(1) Function

This calculates the insertion loss, bandwidth (three), ripple and center frequency of bandwidth 1 within the specified waveform memory. Input/output both use the system variable FI1(*)/FO1(*).

(2) Style

Numerical value type variable name = `FILTER1 (N)`

Waveform memory designation

Dummy

Measuring format: The data measured by LOGMAG (logarithmic amplitude) constitute the execution target.

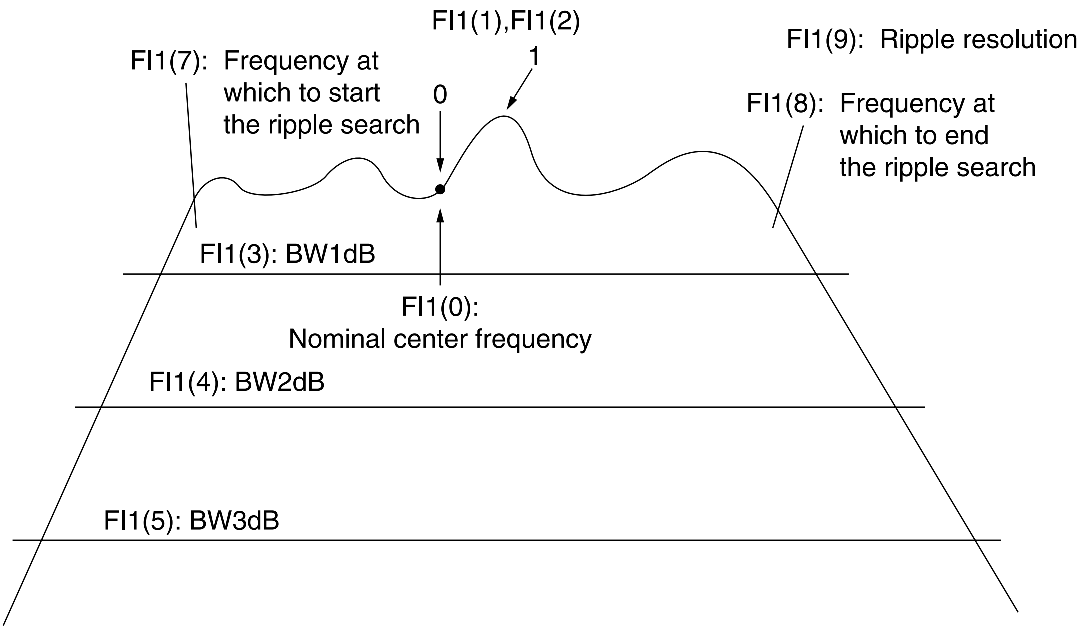
FI1 ( ), FO1 ( ): Numerical value type system array variables (the initial value set at zero)

FI1 (0)	This specifies the nominal center frequency (Hz) of the filter. Numerical value type constant	Input
FI1 (1)	This specifies the reference used to calculate the insertion loss. 0: Value measured of the nominal center frequency 1: Maximum value measured	Input
FI1 (2)	This specifies the reference for bandwidth. 0: Nominal center frequency point 1: Maximum point of value measured	Input
FI1 (3)	This specifies the difference 1 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, bandwidth 1 is not	Input
FI1 (4)	calculated.)This specifies the difference 2 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, bandwidth 2 is not calculated.)	Input
FI1 (5)	This specifies the difference 3 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, bandwidth 3 is not calculated.)	Input
FI1 (6)	Unused (Specify zero at all times.)	Input
FI1 (7)	This specifies the frequency at which to start the ripple search. Numerical value type constant	Input
FI1 (8)	This specifies the frequency at which to end the ripple search. Numerical value type constant	Input
FI1 (9)	This specifies the ripple resolution. Numerical value type constant	Input

FO1 (0)	Insertion loss	Output
FO1 (1)	f_L of bandwidth 1	Output
FO1 (2)	f_H of bandwidth 1	Output
FO1 (3)	BW of bandwidth 1	Output
FO1 (4)	f_L of bandwidth 2	Output
FO1 (5)	f_H of bandwidth 2	Output
FO1 (6)	BW of bandwidth 2	Output
FO1 (7)	f_L of bandwidth 3	Output
FO1 (8)	f_H of bandwidth 3	Output
FO1 (9)	BW of bandwidth 3	Output
FO1 (10)	Center frequency of bandwidth 1	Output
FO1 (11)	Ripple	Output

FILTER 1

<Input>

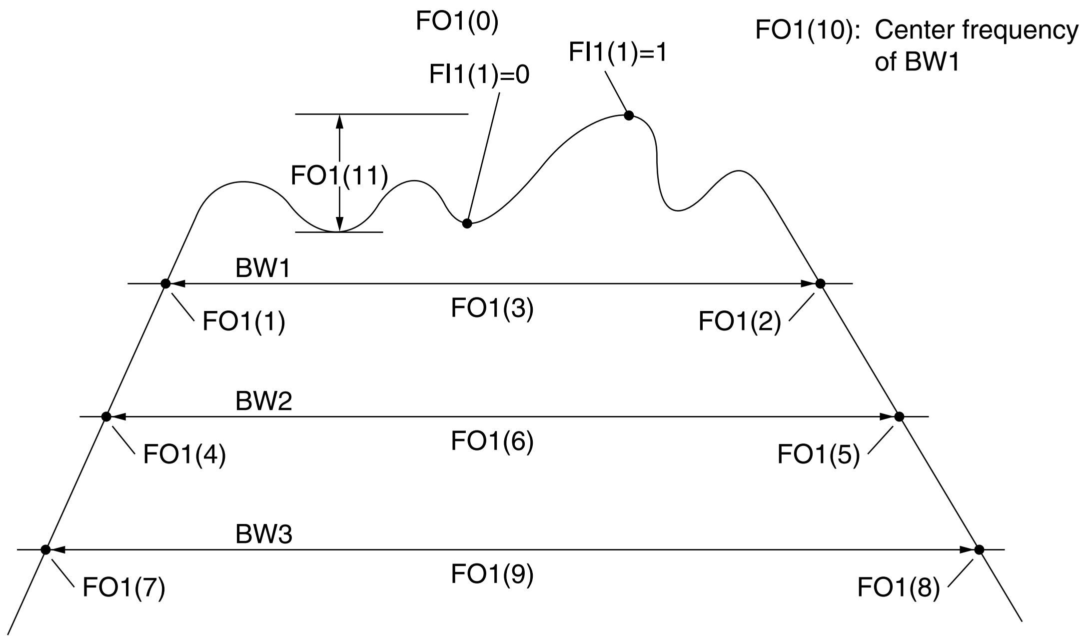


- FI1(1): When the insertion reference equals zero → Output to the insertion loss of the nominal center frequency or F01(0).

  When the insertion reference equals one → Output to the minimum insertion loss or F01(0).
- FI1(2): When the bandwidth reference equals zero → The insertion loss of the nominal center frequency serves as the reference for bandwidth calculation.

  When the bandwidth reference equals one → The minimum insertion loss serves as the reference for bandwidth calculation.
- When the bandwidth, BW1, equals 3 dB, set FI1(3) at 3. This also applies to FI1(4) and FI1(5). When FI1(3) equals zero, the bandwidth is not calculated.

<Output>



5.3.10 FILTER 2 Function

(1) Function

This calculates the maximum DELAY, minimum DELAY and maximum DELAY-minimum DELAY within the specified range of the specified waveform memory, as well as the DELAY of the nominal center frequency.

(2) Style

Numerical value type variable name = `FILTER2(N)`

Dummy

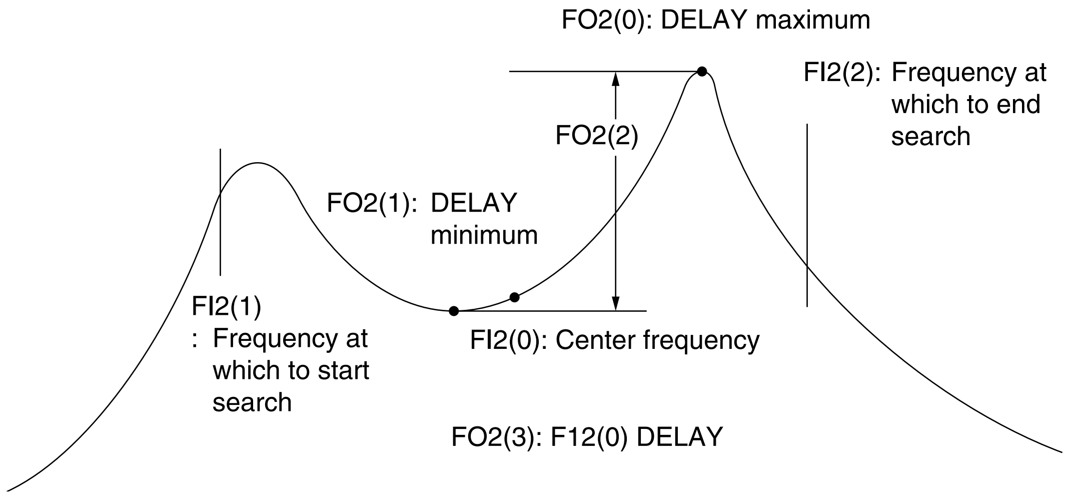
Waveform memory designation

Measuring format: The data measured by DELAY (group delay) constitute the execution target.

FI2 ( ), FO2 ( ): Numerical value type system array variables (the initial value set at zero)

FI2 (0)	This specifies the nominal center frequency. Numerical type constant	Input
FI2 (1)	This specifies the frequency at which to start search. Numerical type constant	Input
FI2 (2)	This specifies the frequency at which to end search. Numerical type constant	Input
FO2 (0)	Maximum DELAY	Output
FO2 (1)	Minimum DELAY	Output
FO2 (2)	Maximum-Minimum DELAY	Output
FO2 (3)	DELAY of the nominal center frequency	Output

N: Same as the FILTER 1 function.



5.3.11 FILTER 3 Function

(1) Function

This calculates the bandwidth (three) in the specified waveform memory as well as maximum measuring value (five) within a specified range.

(2) Style

```
Numerical value type variable name = FILTER3(N)
      |                                     |
      |                          Waveform memory designation
    Dummy
```

Measuring format: The data measured by LOGMAG (logarithmic amplitude) constitute the execution target.

FI3 ( ), FO3 ( ): Numerical value type system array variables (the initial value set at zero)

FI3 (0)	This specifies the reference for bandwidth. 0: Nominal center frequency point 1: Maximum point of value measured	Input
FI3 (1)	This specifies the difference 1 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FI3 (2)	This specifies the difference 2 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FI3 (2)	This specifies the difference 3 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FI3 (4)	This specifies frequency 1 at which to start search. Numerical value type constant	Input
FI3 (5)	This specifies frequency 1 at which to end search. Numerical value type constant	Input
FI3 (6)	This specifies frequency 2 at which to start search. Numerical value type constant	Input
FI3 (7)	This specifies frequency 2 at which to end search. Numerical value type constant	Input
FI3 (8)	This specifies frequency 3 at which to start search. Numerical value type constant	Input
FI3 (9)	This specifies frequency 3 at which to end search. Numerical value type constant	Input
FI3 (10)	This specifies frequency 4 at which to start search. Numerical value type constant	Input
FI3 (11)	This specifies frequency 4 at which to end search. Numerical value type constant	Input
FI3 (12)	This specifies frequency 5 at which to start search. Numerical value type constant	Input
FI3 (13)	This specifies frequency 5 at which to end search. Numerical value type constant	Input

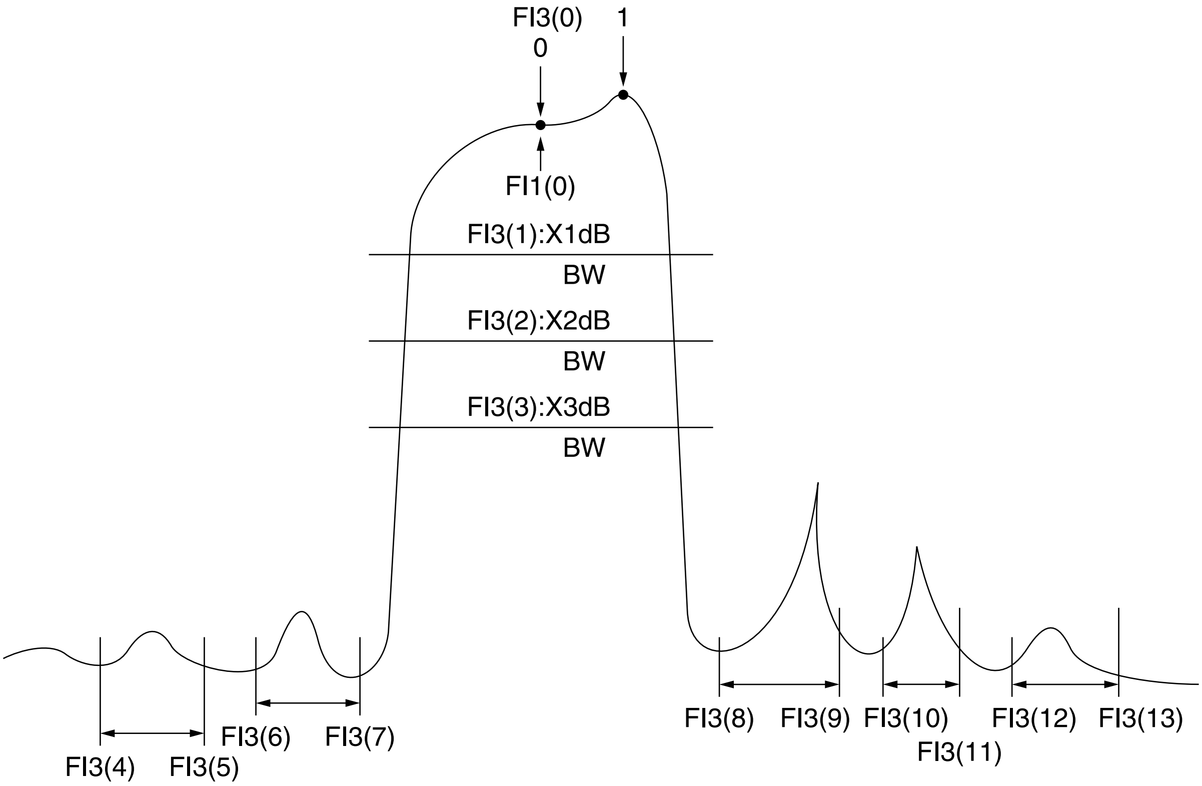
FO3 (0)	Bandwidth 1	Output
FO3 (1)	Bandwidth 2	Output
FO3 (2)	Bandwidth 3	Output
FO3 (3)	Maximum value measured 1	Output
FO3 (4)	Maximum value measured 2	Output
FO3 (5)	Maximum value measured 3	Output
FO3 (6)	Maximum value measured 4	Output
FO3 (7)	Maximum value measured 5	Output

N: Same as FILTER 1 function.

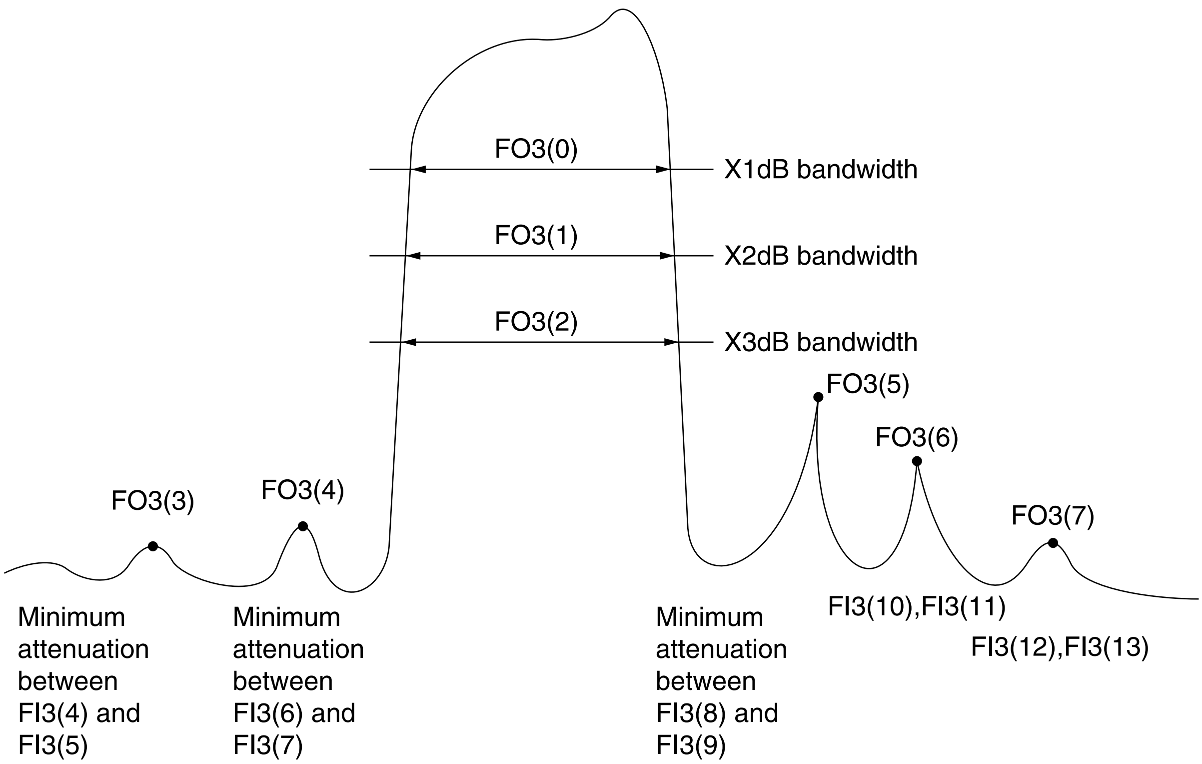
The nominal center frequency is specified by FI1(0).

FILTER 3

<Input>



<Output>


FO3(0)
X1dB bandwidth
FO3(1)
X2dB bandwidth
FO3(2)
X3dB bandwidth
FO3(5)
FO3(6)
FO3(3)
FO3(4)
FO3(7)
Minimum attenuation between FI3(4) and FI3(5)
Minimum attenuation between FI3(6) and FI3(7)
Minimum attenuation between FI3(8) and FI3(9)
FI3(10),FI3(11)
FI3(12),FI3(13)

5.3.12 FILTER 4 Function

(1) Function

This calculates the bandwidth (three) within the specified waveform memory.

(2) Style

Numerical value type variable name = `FILTER4(N)`

Specifled waveform memory

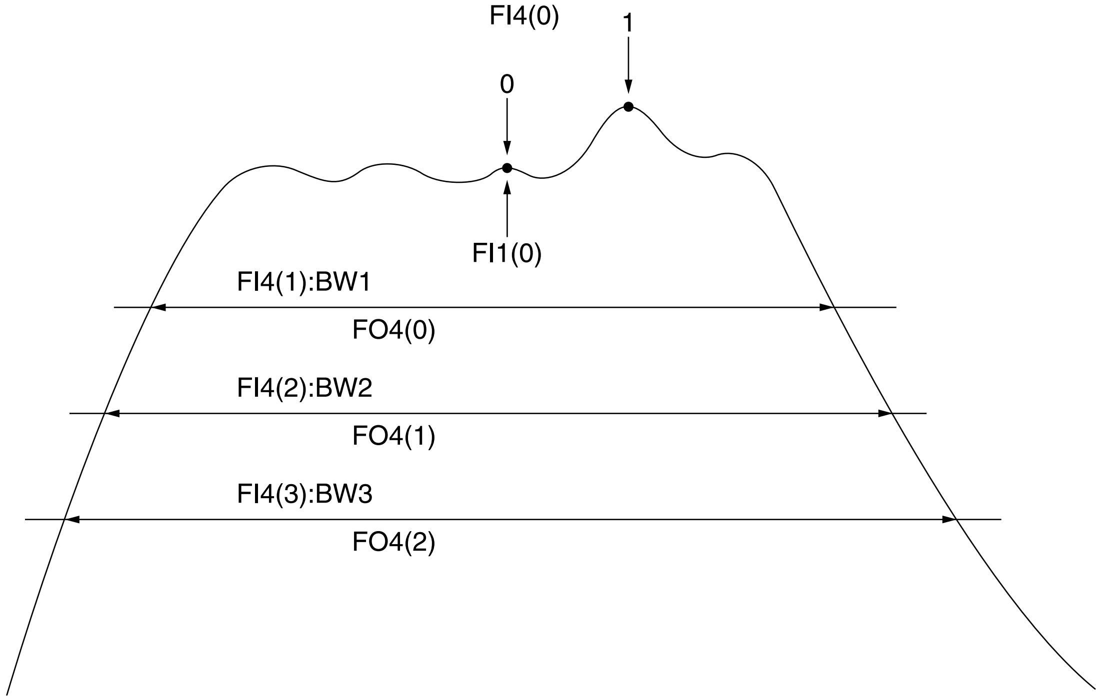
Dummy

Measuring format: The data measured by LOGMAG (logarithmic amplitude) constitute the execution target.

FI4 ( ), FO4 ( ): Numerical value type system array variables (the initial value set at zero)

FI4 (0)	This specifies the reference for bandwidth. 0: Nominal center frequency point 1: Maximum point of value measured	Input
FI4 (1)	This specifies the difference 1 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FI4 (2)	This specifies the difference 2 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FI4 (3)	This specifies the difference 3 from the reference value to calculate the bandwidth. Numerical value type constant (When the difference is zero, the bandwidth is not calculated.)	Input
FO4 (0)	Bandwidth 1	Output
FO4 (1)	Bandwidth 2	Output
FO4 (2)	Bandwidth 3	Output

N: Same as FILTER 1 function.
The nominal center frequency is specified by FI1(0).


FI4(0)
1
0
FI1(0)
FI4(1):BW1
FO4(0)
FI4(2):BW2
FO4(1)
FI4(3):BW3
FO4(2)

5.3.13 FILTER 5 Function

(1) Function

This calculates the maximum measuring value (five) within the specified range of the specified waveform memory.

(2) Style

Numerical value type variable name = `FILTER5(N)`

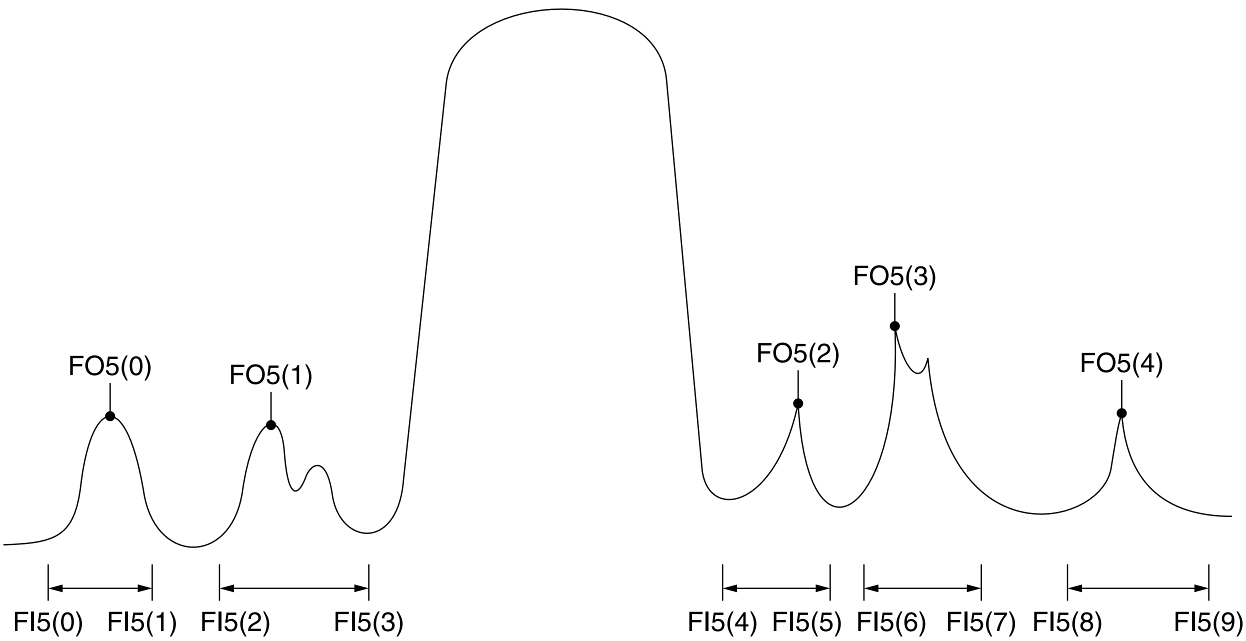
Dummy — Numerical value type variable name

Specified waveform memory — N

Measuring format: The data measured by LOGOMAG (logarithmic amplitude) constitute the execution target.

FI5 ( ), FO5 ( ): Numerical value type system array variables (the initial value set at zero)

FI5 (0)	This specifies frequency 1 at which to start search. Numerical value type constant	Input
FI5 (1)	This specifies frequency 1 at which to end search. Numerical value type constant	Input
FI5 (2)	This specifies frequency 2 at which to start search. Numerical value type constant	Input
FI5 (3)	This specifies frequency 2 at which to end search. Numerical value type constant	Input
FI5 (4)	This specifies frequency 3 at which to start search. Numerical value type constant	Input
FI5 (5)	This specifies frequency 3 at which to end search. Numerical value type constant	Input
FI5 (6)	This specifies frequency 4 at which to start search. Numerical value type constant	Input
FI5 (7)	This specifies frequency 4 at which to end search. Numerical value type constant	Input
FI5 (8)	This specifies frequency 5 at which to start search. Numerical value type constant	Input
FI5 (9)	This specifies frequency 5 at which to end search. Numerical value type constant	Input
FO5 (0)	Maximum value measured 1	Output
FO5 (1)	Maximum value measured 2	Output
FO5 (2)	Maximum value measured 3	Output
FO5 (3)	Maximum value measured 4	Output
FO5 (4)	Maximum value measured 5	Output


FO5(0)
FO5(1)
FO5(2)
FO5(3)
FO5(4)
FI5(0)
FI5(1)
FI5(2)
FI5(3)
FI5(4)
FI5(5)
FI5(6)
FI5(7)
FI5(8)
FI5(9)

5.3.14 GPDLY Function

(1) Function

This function is used to calculate the group delay time at the central frequency from the measurement data (magnitude of vector) and the measurement frequency.

(2) Format

Numerical variable=GPDLY (P0, P1, CH1, [,E])

- P0: Measurement point (lower frequency)
- P1: Measurement point (higher frequency)
- CH1: Measurement channel (1: measurement channel 1: 1/2: measurement channel 2)
- E: Specification of decimal place in function result (0 to -12)

Procedure of calculation

1. Use the measurement and frequency data for the measurement channel designated by CH.
2. Read the measured values (magnitude of vector) at the measurement points designated by P0 and P1.
3. Convert each of the measured values to a magnitude of phase. $\rightarrow \theta\,0, \theta\,1$ (deg)
4. Read the measurement frequencies at the measurement points designated by P0 and P1. $\rightarrow$ f0, f1 (Hz)
5. The value obtained in the following calculation is defined as the group delay time.
   $-(\theta\,1-\theta\,0)/\{360 \times (f1-f0)\} \rightarrow \tau$
6. Using the value designated by E, perform the following calculation and store the result in a numerical variable.
   $\tau \times 10^{E} \rightarrow$ Stored in a numerical variable.
   The assignment of E may be omitted. In this case, E=0 is assumed for calculation.

Notes:

- As the measurement data to be converted into magnitudes of phase, use the vector data in the complex measurement memory. Since this data is not yet processed according to the measurement format such as LOGMAG or PHASE, the result of this function does not depend on the measurement format. However, the measurement data compensation or manipulation due to the X-S, SMOOTHING, and SUB TRACE processing performed after the data processing according to the measurement format is not reflected in this vector data.
- If the frequency difference (f1 – f0) is 0 Hz, 0 is returned as the function result.
- If the value of $(\theta\,1 - \theta\,0)$ is larger than 180 deg, or smaller than -180 deg, subtract or add 360 deg from/to it respectively and assume the result as the phase difference.

5.3.15 GST Function

(1) Function

This serializes the port of the device connected to the GPIB interface and reads out the status byte value.

(2) Style

```
Numerical variable=GST(G)
```

Device GPIB address (0 to 30)

- Effective when the GPIB port, functioning as the controller port, is in the active status.

Section 6
Remote Control Command Used by PTA Program

6.1 Overview

The remote commands to control the main unit are sent within the PTA program, using the PUT statements and WRITE 1,000 statements. Moreover, the GET statements, COM statements and READ 1,000 statements are used to download the measurement parameters and measurement results from the main unit. The control commands defined by the measuring instrument and almost all the inquiry commands can be used as the remote control commands here. The remote control commands prepared for the PTA program can also be used.

6.2 Remote Control Command for PTA

The PTA program sends remote control command style messages using WRITE 1,000 statements and READ 1,000 statements to set up the parameters on the measuring instrument unit as well as download such data from the same unit. PTA has remote control commands exclusively for itself in addition to the remote control commands for the measuring instrument.

Function		Command	Operation
Port switching	Control	PORT␣1	This selects GPIB as the active port.
		PORT␣2	This selects RS-232C as the active port.
		PORT␣3	This selects parallel (Centronics) as the active port.
	Request	PORT?	This requests the current active port.
PTL mode		PTL␣1 or CHR$(4)	This prompts switching to the mode where the external PC executes the command and statement targeting the PTA program and where the PTA program is transferred (PTL transfer mode).
PTL transfer mode		PTL␣2	This prompts switching to the mode where the PTA program is transferred to the external PC (PTL transfer).
PTL mode cancellation		PTL␣Ø or CHR$(3)	This cancels the PTL and PTL transfer modes.
PTA ON/OFF		PTA␣1	This turns PTA ON.
		PTA␣Ø	This turns PTA OFF.

What is PTL mode?

The PTL mode is an operation's mode for communications between the external PC and the PTA of the device. Switching to this mode means that the commands and character rows entered from the external PC are in principle interpreted as the PTA commands and statements. Therefore, the remote commands normally used for measuring control cannot be used. However, the remote command for PTL mode cancellation can be used.

Running and resetting the PTA program is possible in the PTL mode (stopping the program is not possible, though.)

CAUTION ⚠

- **See Chapter 4 Measuring Parameter Setup (1,000 PUT and 1,000 WRITE Statements), and Measurement Parameter/Data Downloading (1,000 GET, 1,000 COM and 1,000 READ Statements)(1), for more details of 1,000 WRITE and 1,000 READ statements.**
- **The WRITE, READ, LISTG, SAVEG and LOADG statements and other GPIB statements, all handled by the PTA program, are effective on ports selected for control by the control port switching function. However, this is not the case when these statements are executed as a result of the direct port designation.**
- **The port designated by the port switching command is not initialized when the PTA program is turned OFF.**

6.3 Method of Program Transfer Between External Personal Computer and PTA

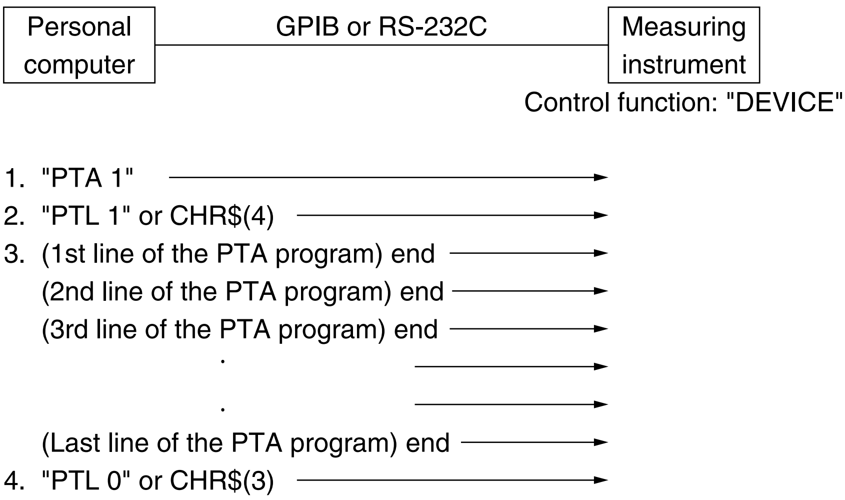
Transferring the PTA program created and edited by the external personal computer to the PTA program memory mounted on the main unit is enabled using GPIB or RS-232C interface.

It is also possible to transfer the PTG program created and edited on the main unit to the external personal computer using GPIB or RS-232C interface.

The ASCII (text) format programs constitute the transfer target.

The section below explains the method of transfer in two reversed directions.

6.3.1 Procedures for Transferring PTA Program from External Personal Computer to Measuring Instrument



Preparations

- Connect the personal computer and the measuring instrument with GPIB (RS-232C).
- Set the control function of the interface on the measuring instrument at “DEVICE.”
- Set up other GPIB (RS-232C) interface conditions.

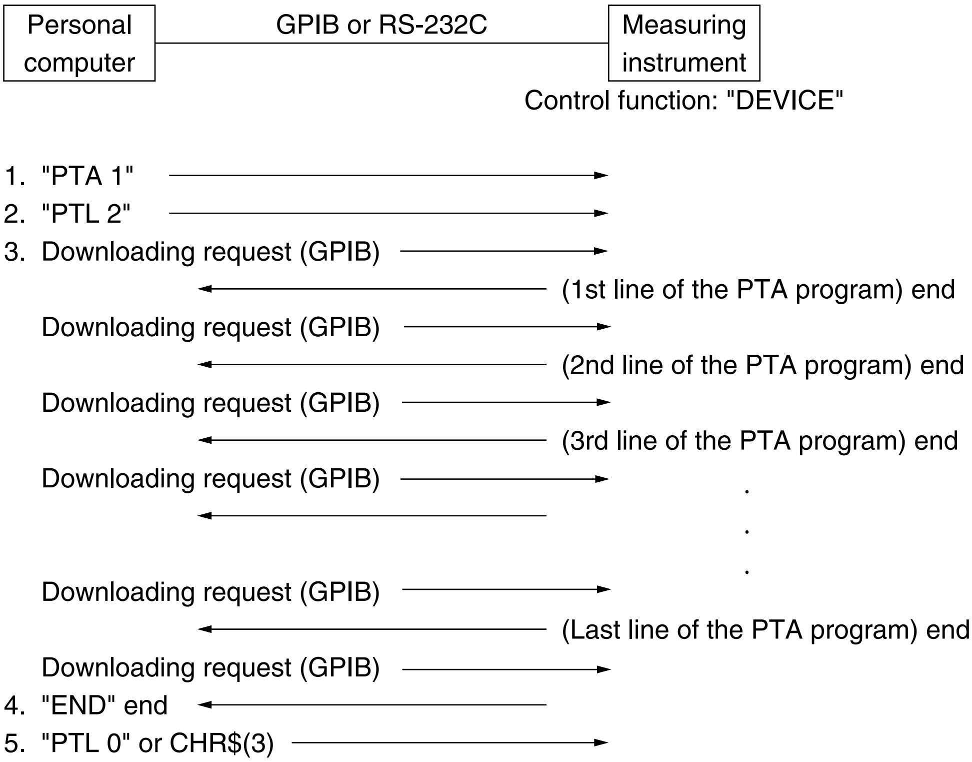
Execution of transfer

(1) Turn ON PTA on the measuring instrument (PTA 1).
(2) Turn ON the PTL mode. (PTL 1 or CHR$(4))
(3) Transfer the PTA program by line.
(4) Cancel the PTL mode when the transmission of the last line is completed. (PTL 0 or CHR$(3))

Note:

The remote commands entered from the external equipment while the measuring instrument is in the PTL mode are interpreted as the commands and statements to be entered to PTA. Therefore, commands such as those used to control the measuring unit result in an error.

6.3.2 Procedures for Transferring PTA Program of Measuring Instrument to External Personal Computer



Preparations

- Connect the personal computer and the measuring instrument with GPIB (RS-232C).
- Set the control function of the interface on the measuring instrument at “DEVICE.”
- Set up other GPIB (RS-232C) interface conditions.

Execution of transfer

(1) Turn ON PTA on the measuring instrument (PTA 1).
(2) Turn ON the PTL mode. (PTL 2)
(3) Send the request to download the PTA program to the measuring instrument. (In this case GPIB)
One line equivalent portion of the program is sent from the measuring instrument (PTA) to the computer. This process is repeated.
(4) The character row of “END” is output finally from the measuring instrument (PTA) to the computer.
(5) The reception of the character row of “END” prompts cancellation of the PTL mode. (PTL 0 or CHR$(3))

Section 7 External Interface at PTA

7.1 Overview

This unit comes with the standard GPIB interface, as well as the following options; RS-232C interface (option 02) and parallel (Centronics) interface (option 02). These external interfaces can be controlled from PTA.

7.2 Selection of Interface Port Controlled by PTA

The interface port controlled by PTA is the same as the active port of the main unit function. The active port can be selected in the “ACTIVE PORT for HARD COPY” item located in the window developed from any one of the menu items, F1: GPIB, F2: RS-232C and F4: HARD COPY, displayed by pressing the LOCAL key or SHIFT, + Copy (Control) key. Switching the control function of the selected interface port into “CONTROLLER” is also required. Execute the CALL IFC sub-routine on PTA to switch the GPIB interface into the “CONTROLLER” status. No command is available to enable PTA to switch the RS-232C interface into the “CONTROLLER” status.

Moreover, the "PORT" remote control command can set up the active port.

- `PORT␣1` Sets the active port at the GPIB port.
- `PORT␣2` Sets the RS-232C interface as the active port.
- `PORT␣3` Sets the parallel (Centronics) interface as the active port.

7.3 GPIB Function at PTA

7.3.1 Functions as the Controller

(1) Program list output and program output

This lists up the program from the ACTIVE port currently selected by the LISTG command to the external printer. The SAVEG command also outputs the PTA program to other measuring instruments.

(2) IFC sending

`CALL␣IFC` is used to send “Interface Clear” to the devices connected on GPIB.

(3) Status byte reading

`GST (G)` command is used to download the status byte of other devices.

(4) Data transmission

The WRITE statement is used to transmit data to the devices on GPIB.

```
WRITE␣M, variable [:Format] [, Variable [:Format] ····]
```

M: External device address (numerical value constant or numerical value variable used)

variable [:Format] [, Variable [:Format] ····]: Output data (character row allowed)

CAUTION ⚠

When M equals 1,000, the main unit functions are set up. In this case, both controller and device modes enable operations.

(5) Data reception

The READ statement is used to receive data from the devices on GPIB.

READ␣M, Variable [, Variable ····]

The data received is entered to substitute the variable.

External device address (numerical value constant or numerical value variable used)

CAUTION ⚠

When the GPIB port is functioning as the device port, the WRITE and READ statements enable dual port memory access.

CAUTION ⚠

Specifying a one-digit or a two-digit figure (such as 5 or 17) as the address means that the control is exercised on the devices connected to the port designated by the PORT command functioning as the GPIB command. On the other hand, specifying a three digit figure (such as 105 and 217) as the address means that the first digit indicates the port No. and the last two digits indicate the address of the device connected to the port No. indicated by the first digit. The former is called the indirect port designation, while the latter is called the direct port designation. Although the address set in accordance with the indirect port designation and the last two digits signifying the address set in accordance with the direct port designation have no meaning to RS-232C, specifying them is necessary as a matter of formality.

Example:

WRITE␣5, "ABC" .... Data sending to address 5 on the active port.

WRITE␣105, "ABC" .... Data sending to address 5 on port No. 1 (GPIB) (Direct port designation)

READ␣217, AS .... Data entry from the address 17 (dummy) on port No. 2 (RS-232C) (Direct port designation)

The address designation explained in the following section is effective on the WRITE, BWRITE, WWRITE, READ BREAD, WREAD, LISTG and SAVEG statements.

The table below shows the relation between the port designation command and active port.

	Indirect port setup	Direct port setup	
	`WRITE 5`	`WRITE 1Ø5`	`WRITE 2Ø5`
After "PORT ␣ 1" execution	GPIB port set as the active port	GPIB port set as the active port	RS-232C port set as the active port
After "PORT ␣ 2" execution	RS-232C port set as the active port	GPIB port set as the active port	RS-232C port set as the active port

Note 1:

Although the address designation has no meaning to RS-232C, specify the address as a matter of formality.

Note 2:

When a device port is specified, writing in/reading from the dual port memory are allowed. In this case, the BWRITE, WWRITE, BREAD, WREAD, LISTG and SAVEG commands and statements cannot be used.

(6) Time limit

The time limit is set at 20 seconds (initial value). Use the GPIB commands as shown below to modify the time limit.

`GTM ␣ t` 0 - 99999 (Unit: second)

Setting t at zero means absence of the preset time limit (indefinite waiting).

(7) READ/WRITE Terminator

The GPIB terminator conditions are set as follows.

Terminator for talker (sending)

<Port> instructions	Terminator code
< GPIB > WRITE LISTG	CR + LF or LF, (either one will do.) (Follow the TRM command.)

* TRM command constitutes a part of the GPIB commands.

`TRM ␣ 1`(CR + LF)
`TRM ␣ Ø`(LF only)

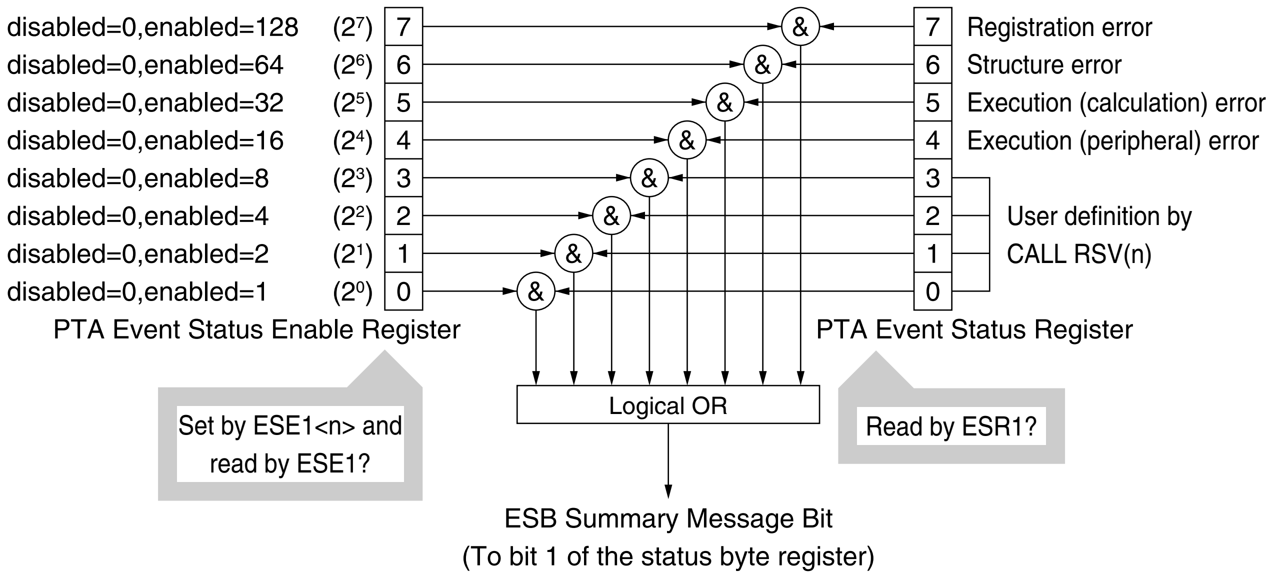
Terminator for listener (receiving)

<Port> instructions	Terminator code
< GPIB > READ	LF or CR + LF (either one will do.)

7.3.2 Functions as the Device

(1) Service request sending

CALL␣RSV (M) is used to send a command to request the external controller for service.



Bit	Event name	Explanations
7	Registration error	Error during program registration
6	Structure error	Error in the program structure
5	Execution (calculation) error	Error in the calculation during program execution
4	Execution (peripheral) error	Peripheral error during program execution
3	User definition event	Event the user is allowed to set at user's own will by CALL RSV (n)
2	User definition event	
1	User definition event	
0	User definition event	

7.4 Parallel (Centronics) Funciton at PTA

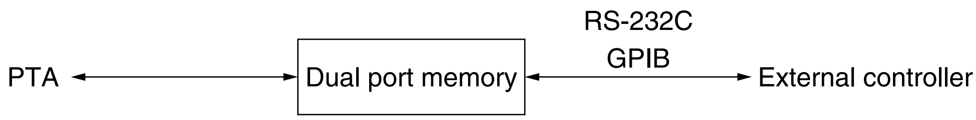
(1) Program list

The LISTG command is used to output the list of programs from the parallel (Centronics) port to the external printer. In this case, the parallel (Centronics) port must have been specified as the active port.

7.5 Dual Port Memory

(1) Application and structure

The dual port memory is built in PTA and enables, in a flexible manner, PTA to save and download data as well as external controller to save and download data. The dual port memory functions as a route to output the data and measurement results gained from the PTA program to the external controller, and at the same time, is used for the purpose of achieving communications between PTA and external controller.



The dual port memory comprises 32 memory block of 32 byte each in capacity. Accessing each memory block unit requires designation of the memory No. Memory Nos. 0 - 31 can be designated.

Dual Port Memory Structure

Memory No. 0	32 byte
Memory No. 1	32 byte
Memory No. 2	32 byte
.	.
.	.
.	.
Memory No. 30	32 byte
Memory No. 31	32 byte

(2) Saving data to the dual port memory

Style

- Saving from PTA

  `WDPM` Memory No., Data to be saved or

  `WRITE` Memory No., Data to be saved

- Saving from external controller

  "`PMY` Memory No., Data to be saved"

- Make sure to designate the memory No. without fail before saving data to the dual port memory. Moreover, writing of data starts from the header byte of the memory block which matches the specified memory No.
- The ending code (LF) of one byte is added at the end of the data to be saved.
- When the size of the data to be saved exceeds 32 bytes, saving of the data can stretch to the next memory. Also, when the size of the data to be saved comes in 32 bytes just, the ending code is stored at the header of the next memory.

- Trying to save data of the size which goes beyond the final byte of the final memory results in an error, and the saving of the data fails. In this case, data saved before are maintained.
- The data are always stored in the ASCII format on the memory. When the data are saved from PTA, the storage size varies depending on the data type, as shown below.

(a) For character constant/variable

- Saved in the ASCII format with one character set at one byte.
- Trying to save data using the character variable without format designation results in saving of (number of bytes equivalent to the array size) + (1 byte: space code). The ending code is saved at the end.
- Trying to save data using the character variable with uppercase format designation results in saving of one-byte worth of space code at the end of the data. The ending code is saved at the end.
- Trying to save data using the character constant results in saving of data equivalent to the number of characters in the " ". The ending code is saved at the end.

(b) For numerical value variable

- The numerical value is converted into the character row (ASCII data) and the data equivalent to the size of the character row is saved. The minus and decimal point symbols take up one-byte worth of size each. The ending code is saved at the end.

(c) For bit variable

- With 1 byte/1 bit, the numerical value of 0/1 of each bit is converted into the character row (ASCII data) and the data equivalent to the size of the character row is saved.
- The style of saving of the data with/without format designation is the same as that applicable to the case using the character variable.
- The BWRITE and WWRITE statements cannot be used.

Example:

- Saving from PTA
  `WDPM Ø, "MEASEND"` : Save "MEASEND" in memory No. 0.
- Saving from the external controller
  `"PMY Ø, MEASSTART"` : Save "MEASSTART" in memory No. 0.

Notes:

- The WDPM statement is used exclusively for the purpose of saving data to the dual port memory.

(3) Data downloading from the dual port memory

Style

- Downloading from PTA

  `RDPM` memory No., Input variable [, Input variable ..] or

  `READ` Memory No. , Input variable [, Input variable]

- Downloading from the external controller

  "`PMY?` Memory No. from which to start downloading, Number of memory units" + Downloading command

- Make sure to designate the memory No. without fail to download the data from the dual port memory. In principle, data output must be in a series leading up to the ending code (LF). However, when the data is downloaded up to the final byte of the last memory block, the data downloading is considered to have terminated.
- Trying to download the data saved over multiple memory units by specifying the memory No. located in between first and last memory units where data saving begins and ends respectively results in downloading of data from in between the data header and end.
- Data downloading from PTA in principle involves the data segment which ends with the ending code. However, when the ", " exists in data, the ", " is considered to signal the data demarcation. Therefore, the data saved into the input variable are up to the point prior to ",". In this case, specify multiple input variables. The discrepancy between the number of data segments and the number of input variables may result in a downloading error (when the number of input variables is greater than the number of data segments) or result in a situation where the data to be output remain within PTA (when the number of input variables is smaller than the number of data segments).
- Use the "PMY?" command to download data from the external controller. The "PMY?" command enables designation of the memory No. from which to start downloading and the number of memory unites to be downloaded. In this case, the data segment which starts from the header of each memory block. and ends at the ending code is output as one set of data.
- Substituting a data piece in the dual port memory for the input variable may in some cases result in a downloading error, because of the discrepancy in the data type between the substituting data and the input variable.
- The BREAD and WREAD statements cannot be used.

Example:

- Downloading from PTA

  `RDPM Ø, A$` : Download data from memory No. 0 and save them as character variable A$.

- Downloading from the external controller

  `"PMY? Ø, 3"` : Request output of three-memory unit worth of data starting from memory No. 0 (memory Nos. 0, 1 and 2). Following this, data is downloaded three times.

Note:

The RDPM statement is used exclusively for the purpose of downloading data from the dual port memory.

(4) Details of writing into/reading from the dual port memory

Control demand from the external controller	Dual port memory contents	
"PMY␣Ø,ABC"	Memory 0	ABC(LF)
"PMY␣1,123"	Memory 1	123(LF)
"PMY␣2,XYZ"	Memory 2	XYZ(LF)

The execution statements on the left above correspond to the dual port memory contents shown on the right.

- Comma <,> in the dual port memory

  The data output command comes in segments, each running from the header of the memory No. designated to the ending code. A comma <,> used is considered a part of the data segment in this case. On the other hand, as the READ statement of PTA and the controller demarcates data segments using the comma <,> to substitute data for the variable.

Dual port memory contents

Memory 0	ABC,DEF(LF)
Memory 1	XYZ(LF)

Execute the following statements to read the contents of the dual port memory Nos. 0 and 1 shown above.

```
RDPM Ø,A$,B$
```

In this case, A$ is substituted by ABC. As the comma <,> in the data functions as the separator, DEF substitutes for B$.

Section 8 PTA Error Message

An error message appears when an error is encountered in the PTA command and the program. Errors are classified into two categories; fatal errors which preclude execution and warning errors which enables continuation of execution.

- Error to preclude execution (F: Fatal error) : This terminates execution of the program under any conditions.
- Error to enable continuation of execution (W: Warning error) : This terminates execution of the program when no error statement exists on the line following the line where an error is encountered. However, when an error statement exists, the execution of the program can continue. Moreover, the error interruption processing enables continuation of the program execution.

8.1 ERROR Message Style

The error message appears in the style shown below.

ERROR Error level Error No. [, Error generation line No.]

Displayed during the program execution

8.2 ERROR Statement

(1) Function

In case of a warning error encountered during the execution of the program, the ERROR statement enables continuation of the program execution.

(2) Style

```
ERROR(21Ø,1ØØØ)
```

Line of the program executed next

Error No.

This statement means that when the error encountered on the previous line corresponds to error No. 210, 1,000-line worth of the program can be executed.

Discordance with the error No. prompts an error message to appear to terminate the program execution.

(3) Example

```
1Ø   X=Ø
2Ø   Y=1ØØ/X
3Ø   ERROR (21Ø,1ØØ)    ; This prompts jump to line 100 in case of an error having been encountered
                          indicating that the divisor is zero.
4Ø   Y=Y+{5Ø
        :
```

8.3 ERRMAIN Statement

(1) Function

In case of a warning error encountered during the execution of the program, the ERRMAIN statement enables branching to the main routine.

(2) Style

`ERRMAIN` (Error No.)

Example:

```
10    INPUT A
20    GOSUB 1000
30    :
      :
1000  WRITE 217, A
1010  ERRMAIN (222)    ;  This prompts return to the main routine in case of an error having been encountered indicating that the WRITE statement is unable to output the data.
1020
```

Note:

Executing the ERRMAIN statement in the routine which comes in the uppermost hierarchy results in error No. F213.

8.4 Error Processing Sub-Rutine

8.4.1 ON ERROR Statement

(1) Function

This registers the sub-routine for branching (interruption) in case of an error happening.

(2) Style

`ON ERROR` Line No. (or * label)

When a warning error is encountered during the execution of the instructions, an interruption is generated to prompt execution of the error processing sub-routine from the line No. (or label).

8.4.2 OFF ERROR Statement

(1) Function

This cancels the registration of the sub-routine for branching (interruption) in case of an error happening.

(2) Style

`OFF ERROR`

After executing the instructions, the error interruption is not generated.

8.4.3 Return from Error Processing Sub-Routine (RETERR, RETRY, RESUME, GIVE UP Statement)

(1) Function

This enables return from the error interruption.

(2) Style

`RETERR`	(Continuation from the statement following the statement which has generated an error)
`RETRY`	(Continuation following the retry of the statement which has generated an error)
`RESUME`	(Continuation from the line specified)
`GIVEUP`	(Suspension of the program execution)

Note:

See Chapter 4 RETERR Statement and GIVE UP Statement for more details(1).

8.4.4 Function ERRREAD (m)

(1) Function

This enables downloading of the error code or the line generating an error in the error processing sub-routine.

(2) Style

```
V=ERRREAD(Ø)        (Downloading of the error code)
V=ERRREAD(1)        (Downloading of the line generating an error)
```

(3) Example

```
100 ON ERROR 200        ; This prompts jump to line 200 in case of an error having been encountered.
110 INPUT X
120 Y=100/X
130 PRINT Y
140 GOTO 110
150 STOP
200 C=ERRREAD (0)
210 IF C=210 GOSUB 300  ; When the divisor is zero, this prompts the display of "ERROR/0" and the execution continues from line 130.
220 IF C<>210 GIVEUP    ; This suspends the program execution in case of other errors having been encountered.
230 RETERR
300 PRINT "ERROR/0"
310 RETURN
```

8.5 Error List

Table 8-1 shows the error No. and the cause of the error. F (Fatal) and W (Warning) in Table 8-1 refer to an error to preclude execution and an error to enable continuation of execution respectively.

Table 8-1 PTA Error List (1/8)

Error No.	Error Contents	W,F
0	The RETURN[↵] key has been pressed without entering the command or the statement first.	F
1	The number of symbol characters has exceeded the preset limit. The variable name comprises greater than eight characters. The program name comprises greater than six characters.	W
2	Error in the numerical value constant formula Example: `Ø..1 4.5EE2`	W
3	Error in the numerical value constant style The number of digits entered has exceeded the preset limit. The value is too large or small.	W
4	Error in the character constant style Example: `A$="ABC`	W
5	Error in the format Example: `PRINT A:G6.2`	W
6	Error in the command style (the statement cannot be interpreted.) Example: `GOTO ABC`	W
7	The statement description is insufficient. Example: `GOTO`	W
8	The statement description is excessive. Example: `GOTO 1ØØ, 2ØØ`	W
9	The number of invariable digits has exceeded 256. Up to 256 user defined variables can be registered.	W
10	A character which cannot be interpreted exists. Example: `-1ØØ`	W
11	Error in the binary and hexadecimal constant style Example: `8#=# 11Ø`	W
12	Error in the binary and hexadecimal constant numerical value (The numerical value is too large.) Binary constant: Up to eight characters Hexadecimal constant: Up to two characters Example: `8#=#1ØØØØØØØØ`	W

Table 8-1 PTA Error List (2/8)

Error No.	Error Contents	W,F
13	The number of format digit is too large. Example: `PRINT A:F6.5`	W
14	The command operand cannot be interpreted. Example: `LIST A, B`	W
15	The number of command operands is too small. Example: `LISTG`	W
16	The number of command operands is too large. Example: `DELETE 1ØØ,1ØØ,3ØØ`	W
17	The line No. has gone beyond 65535. (The program line Nos. are from 1 to 65535.)	W
20	Too lengthy per-line program disables assembly.	W
21	A label whose line No. has not been defined is used for the command operand.	W
Comment	Error Nos. 0 - 21 are generated during command execution or command entry to the program. However error Nos. 6 to 8 are also generated during statement execution.	

Table 8-1 PTA Error List (3/8)

Error No.	Error Contents	W,F
101	The command operand value is inappropriate. Example: `LIST 1ØØ,1Ø`	F
102	The program capacity has exceeded the memory capacity.	F
103	The line No. or the program specified by the command operand is not found. LIST, LISTG, DELETE, RENUM and SAVE commands	F
104	Too many GOTO and GOSUB statements have disabled RENUM execution. When the number of GOTO and GOSUB statements goes beyond 100, executing RENUM is disabled.	F
105	As the line No. specified by the GOTO and GOSUB operand is not found, executing RENUM is disabled.	F
111	The line No. goes beyond 65535 when the execution of the PCOPY and RENUM commands is tried.	F
Comment	Error Nos. 101 to 105, as well as 111, are generated during command execution.	

Table 8-1 PTA Error List (4/8)

Error No.	Error Contents	W,F
120	The media is write-protected.	W
121	The media is not mounted.	W
122	The media memory has exceeded the limit.	W
123	The specified program is not saved on the media.	W
124	The media is broken.	W
125	The memory type is different.	W
126	The format type is different.	W
127	The media has not been formatted.	W
150	The label has not been defined yet, or defined doubly.	F
151	No DATA statement is found.	F
180	The command sent from PTA to the main measuring unit erred.	W
Comment	Error Nos. 120 to 127 are generated when an error is encountered during the media access by the command or the statement.	

Table 8-1 PTA Error List (5/8)

Error No.	Error Contents	W,F
201	The program cannot resume. CONT command	F
202	The line No. specified is not found. RUN has been executed in the absence of a program. RUN and CONT commands and GOTO and GOSUB statements	W
203	The array value is inappropriate in the DIM statement. The array values are 1 to 1024. (However, the bit type values are 1 to 8, while the character type values are 1 to 255.)	W
204	Used as a simple variable or system variable before being array-declared by the DIM statement.	W
205	Doubly array-declared.	W
206	The variable memory runs short. (Basically, the program memory has exceeded the limit.)	F
207	The character data or the bit data are calculated.	W
208	The data types are combined in a way to reject conversion.	W
209	Overflow or underflow. Input data is out of range.	W
210	The divisor is zero in the calculation.	W
211	The parameter value for the arithmetic function is outside the preset range.	W
212	The sub-routine or FOR and NEXT statement nesting has risen to over 10th power.	F
213	No destination for the RETURN statement return is found.	F
214	The data types on the left and right members of the IF statement come in an incomparable combination.	W

Table 8-1 PTA Error List (6/8)

Error No.	Error Contents	W,F
215	The SOS statement has been executed.	F
216	No corresponding FOR statement is found, meaning in other words too many NEXT statements. RUN and CONT commands and GOTO and GOSUB statements	W
217	The input data type is wrong in the INPUT statement.	W
218	The input data volume runs short in the INPUT statement.	W
219	The input data volume has grown excessive in the INPUT statement. Or too large	W
220	The code on the left member is negative in the power calculation. Example: `-1!I5`	W
221	No data can be entered on GPIB. (The talker device is not connected.)	W
222	No data can be output on GPIB.	W
223	The parameter value in the statement is outside the preset range, or the variable type is inappropriate. Example: `WAIT A$`	W
224	The simple variable has the array character attached.	W
225	The array variable has no array character attached.	W
226	The array character attached to the array variable exceeds the limit declared by the DIM statement. Note) The scope of the character attached declared by DIM J (5) is J (0) to J (4).	W
227	PTA cannot be executed as the device on GPIB.	W
228	PTA cannot be executed as the controller on GPIB.	W

Table 8-1 PTA Error List (7/8)

Error No.	Error Contents	W,F
229	No STOP statement to end the program execution is found.	W
230	A system variable unable to be referenced has been referenced.	W
231	Substituting a value for a system variable which does not allow substitution has been tried.	W
232	The character attached to the array variable is not the numerical value type.	F
233	The parameter in the logical function is not the bit type.	W
234	The parameter in the FOR statement is the character type or the bit type.	W
235	The I/O type designation in the EVENT statement falls outside the preset range of 0 to 99.	W
236	The variable symbol in the NEXT statement and the variable in the FOR statement which comes immediately before the NEXT statement are different. Example: `3Ø FOR C = ···` `9Ø NEXT D`	W
237	Six or more character variables or variables are used in the INPUT, PRINT, READ and WRITE statements. Example: `PRINT"FREQ", F(C), "Hz", "LEVEL", LEV "dBm"`	W
238	The format type and the variable type in the PRINT and WRITE statements are different.	W
239	The operand in the LISTG, WRITE and READ statements falls outside the preset range of 0 to 31. Example: `LISTG 35`	W
240	The variable and constant values in CALL and system function fall outside the preset range.	W
241	The variable and constant types in CALL and system function are inappropriate.	W
242	The system variable is used in CALL and the system function.	W

Table 8-1 PTA Error List (8/8)

Error No.	Error Contents	W,F
243	The RETURN or the RETMAIN statement has been used to execute a return from the event interruption processing or error interruption processing.	F
244	The media data file has not been opened.	W
245	The media data file has already been opened.	W
246	The media data file has already been entirely downloaded.	W
247	The media data type and the variable type cannot be converted into each other.	W
248	The input data volume in the READ statement is excessive, or too large.	W
249	The input data volume in the READ statement runs short.	W
250	The input data type in the READ statement is wrong.	W
251	The RETINT statement has been executed on processes other than the event interruption processing. Or, although the GOSUB statement was executed during the event interruption processing, the RETINT statement was executed prior to execution of the RETURN statement to execute a return.	F
252	The RETERR, RETRY, RESUM and GIVEUP statements have been executed on processes other than the error interruption processing. Or, although the GOSUB statement was executed during the error interruption processing, the aforementioned statements were executed prior to execution of the RETURN statement to execute a return.	F
253	The ERRREAD function was executed on processes other than the error interruption processing.	F
254	The STATUS function was executed on processes other than the event interruption processing.	F

Section 9 PTA Parallel I/O Port Control

9.1 Overview

The Personal Test Language (PTA) enables control of the PTA parallel I/O port (hereafter referred to as the I/O port). Therefore, the auto handler, trimming unit and other units can be readily connected to the I/O port to configure the Personal Test Automation (PTA) system. All the input and output signals to the I/O port adopt the negative logic.

9.2 I/O Port Signal Explanation/Standard

The following section explains the names of individual pins located on the I/O port connector as well as their standards.

■ GND (Pin No. 1)

Ground pin

■ INPUT (Pin No. 2)

Pulse input terminal. The internal flip-flop (F/F) of both OUTPUT 1 and OUTPUT 2 is set by this pulse input. Mainly used for measurement start executed by the external devices, INPUT checks whether the internal F/F is set in the PTA application program, and programs the application to commence the measuring routine when the internal F/F is found set. The system variable EX0 is used to check and reset the internal F/F. The pulse is pushed up to 10 kΩ internally and input at the TTL level.

■ OUTPUT 1 (Pin No. 3)

Latch output terminal. OUTPUT 1 is the output of the internal F/F set by the pulse input to INPUT 1 and PTA application program. Mainly used as the status output which displays the progress of the measurement and data processing, OUTPUT 1 programs the application to reset the internal F/F when the measurement or the data processing is completed. The system variable EX0 is used to set and reset the internal F/F. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 12 mA

■ OUTPUT 2 (Pin No. 4)

Latch output terminal. OUTPUT 2 is the output of the internal F/F set by the pulse input to INPUT 1 and PTA application program. Mainly used as the status output which displays the progress of the measurement and data processing, OUTPUT 2 programs the application to reset the internal F/F when the measurement or the data processing is completed. The system variable EX0 is used to set and reset the internal F/F. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 12 mA

■ Output port A 0 to 7 (Pin Nos. 5 to 12)

Output port A 0 to 7 refers to the latch output terminal, which enables the program to output the 8-bit data. (synchronization signals or strobe signals cannot be output.) The system variable IOA is used to output data to output port A. The signal is pushed up to 10 kΩ internally. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 2.6 mA.

■ Output port B 0 to 7 (Pin Nos. 13 to 20)

Output port B 0 to 7 refers to the latch output terminal, which enables the program to output the 8-bit data. (synchronization signals or strobe signals cannot be output.) The system variable IOB is used to output data to output port B. The signal is pushed up to 10 kΩ internally. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 2.6 mA.

■ I/O port C 0 to 3 (Pin Nos. 21 to 24)

I/O port C 0 to 3 refers to the state input/latch output terminal. The I/O port performs the 4-bit as well as write strobe signal (31 pin) output. The system variable IOC is used to input and output data. The system variable EIO is used to switch the port between the input and output mode. The signal is pushed up to 10 kΩ internally. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 2.6 mA.

■ I/O port D0 to 3 (Pin Nos. 25 to 28)

I/O port D 0 to 3 refers to the state input/latch output terminal. The I/O port performs the 4-bit as well as write strobe signal (31 pin) output. The system variable IOD is used to input and output data. The system variable EIO is used to switch the port between the input and output mode. The signal is pushed up to 10 kΩ internally. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 2.6 mA.

■ Port C status (Pin No. 29)

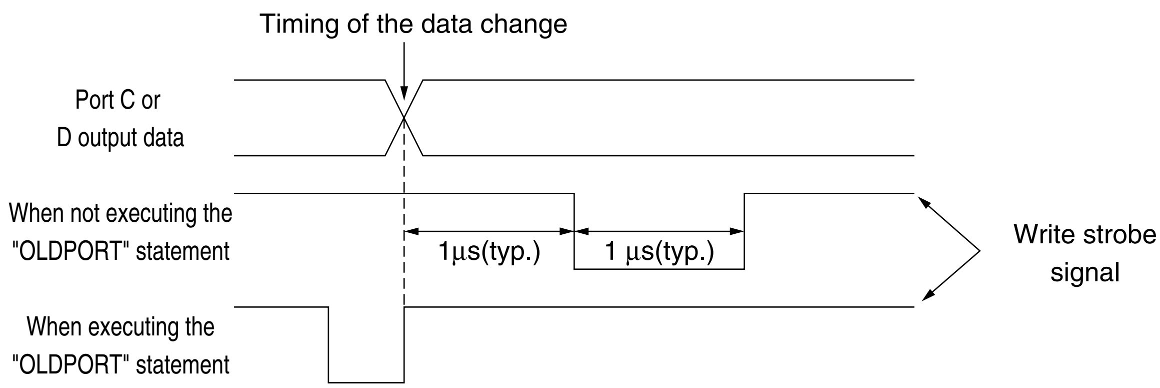
Port C status refers to the status line which displays the mode to external devices. When the system variable EIO switches the I/O port C into the input mode, the status switches to LOW. On the other hand, when the same system variable switches the I/O port C into the output mode, the status switches to HIGH. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 12 mA

■ Port D status (Pin No. 30)

Port D status refers to the status line which displays the mode to external devices. When the system variable EIO switches the I/O port C into the input mode, the status switches to LOW. On the other hand, when the same system variable switches the I/O port C into the output mode, the status switches to HIGH. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 12 mA

■ Write strobe signal (Pin No. 31)

Pulse input terminal. The pulse is output when the data is output from either I/O port C or I/O port D. Executing the “OLDPORT” statement prompts switching of the timing of the write strobe signal pulse generation. Output at the TTL level. Maximum rate IOL = 24 mA, IOH = 12 mA.



■ Interruption signal (Pin No. 32)

Interruption input terminal. Entering the pulse signal to this terminal enables interruption to the PTA hardware. The signal is pushed up to 10 kΩ internally and input at the TTL level.

I/O Port Standard List

Pin No.	Name	Standard	System Variable Name
1	GND	Ground	
2	INPUT 1	TTL level, Negative logic, Pulse Input, Pulse width of 1 µs or greater	EX0
3	OUTPUT 1	TTL level, Negative logic, Latch output	EX0
4	OUTPUT 2	TTL level, Negative logic, Latch output	EX0
5	Output port A0	TTL level, Negative logic, Latch output	IOA
6	Output port A1	TTL level, Negative logic, Latch output	IOA
7	Output port A2	TTL level, Negative logic, Latch output	IOA
8	Output port A3	TTL level, Negative logic, Latch output	IOA
9	Output port A4	TTL level, Negative logic, Latch output	IOA
10	Output port A5	TTL level, Negative logic, Latch output	IOA
11	Output port A6	TTL level, Negative logic, Latch output	IOA
12	Output port A7	TTL level, Negative logic, Latch output	IOA
13	Output port B0	TTL level, Negative logic, Latch output	IOB
14	Output port B1	TTL level, Negative logic, Latch output	IOB
15	Output port B2	TTL level, Negative logic, Latch output	IOB
16	Output port B3	TTL level, Negative logic, Latch output	IOB
17	Output port B4	TTL level, Negative logic, Latch output	IOB
18	Output port B5	TTL level, Negative logic, Latch output	IOB
19	Output port B6	TTL level, Negative logic, Latch output	IOB
20	Output port B7	TTL level, Negative logic, Latch output	IOB
21	I/O port C0	TTL level, Negative logic, State input/Latch output	IOC
22	I/O port C1	TTL level, Negative logic, State input/Latch output	IOC
23	I/O port C2	TTL level, Negative logic, State input/Latch output	IOC
24	I/O port C3	TTL level, Negative logic, State input/Latch output	IOC
25	I/O port D0	TTL level, Negative logic, State input/Latch output	IOD
26	I/O port D1	TTL level, Negative logic, State input/Latch output	IOD
27	I/O port D2	TTL level, Negative logic, State input/Latch output	IOD
28	I/O port D3	TTL level, Negative logic, State input/Latch output	IOD
29	Port C status	TTL level, Input mode: LOW, Output mode: High	EIO
30	Port D status	TTL level, Input mode: LOW, Output mode: High	EIO
31	Write strobe signal	TTL level, Negative logic, Pulse output , Pulse width of 1 µs	
32	Interruption signal	TTL level, Negative logic, Pulse input, Pulse width 1 µs or more	
33	NC		
34	+5 V output	Max. 100 mA	
35	NC		
36	NC		

Note:

NC is a non-contact terminal.

9.3 System Variable to Access I/O Port

Six system variables as shown below are available for the I/O port.

```
EXØ
IOA ()
IOB ()
IOC ()
IOD ()
EIO
```

The section below provides explanations of these system variables. Incidentally, the "setup" and "download" in the explanations mean the data substitution for the system variable and the data downloading from the system variable respectively.

- EX0 .... Used to execute setup of the OUTPUT 1 and OUTPUT 2 status and download of the INPUT 1 status.
  The EX0 variable type is the numerical value type.
  The meaning of the data which execute setup/download targeting EX0 is shown by the table below.

Setup/Download Targeting EX0

Data	Download	Setup
0	Input 1 is reset.	OUTPUT 1 reset (= "H") OUTPUT 2 reset (= "H")
1	Input 1 is set.	OUTPUT 1 set (= "L") OUTPUT 2 reset (= "H")
2		OUTPUT 1 reset (= "H") OUTPUT 2 set (= "L")
3		OUTPUT 1 set (= "L") OUTPUT 2 set (= "L")

When turning power ON, tuning PTA ON or pressing the RESET key, all OUTPUT1, OUTPUT2 and INPUT 1 are reset. INPUT 1 and OUTPUT 1 are interlocked. The OUTPUT 1 status where signals are entered from the external INPUT 1 terminal is set. Resetting OUTPUT 1 switches the INPUT 1 status into the reset status.

- IOA .... Used to output the 8-bit data to output port A. The IOA variable is the bit type.
- IOB .... Used to output the 8-bit data to output port B. The IOB variable is the bit type.
- IOC .... Used to input and output the 4-bit data at I/O port C. The IOC variable is the bit type. The system variable, EIO, switches between the input and output mode.
- IOD .... Used to input and output the 4-bit data at I/O port D. The IOD variable is the bit type. The system variable, EIO, switches between the input and output mode.
- EIO .... Used to set I/O ports C and D either in the input mode or output mode. The EIO variable is the numerical value type. The meaning of the data which execute setup/download targeting EIO is shown by the table below.

Setup/Download Targeting EIO

Data	Setup/Download
0	Port C: Input mode Port D: Input mode
1	Port C: Output mode Port D: Input mode
2	Port C: Input mode Port D: Output mode
3	Port C: Output mode Port D: Output mode

- When the power is launched, PTA is ON, and the RESET key is pressed, both ports C and D switch into the input mode.

9.4 Interrupt by I/O Port

Setting pin No. 32 of the I/O port at LOW causes an interrupt to be applied to PTA. When interrupts by the I/O port are enabled by the ENABLE EVENT statement, branching to the line No. defined by the ON EVENT statement is executed when an interrupt is generated. Moreover, the IOEN and ON IO GOTO (GOSUB) statements enable interrupt processing. See sections 4.2.23 to 4.2.30 for the details of the statements concerning I/O interrupts.

```
10   ON EVENT 41, 1000
20   ENABLE EVENT 41,0,0,0,1
30
                         Measuring routine

GOTO 30
1000 REM"PRINT OUT"
1010 PRINT LMAX, "dB"
1020 RETINT
```

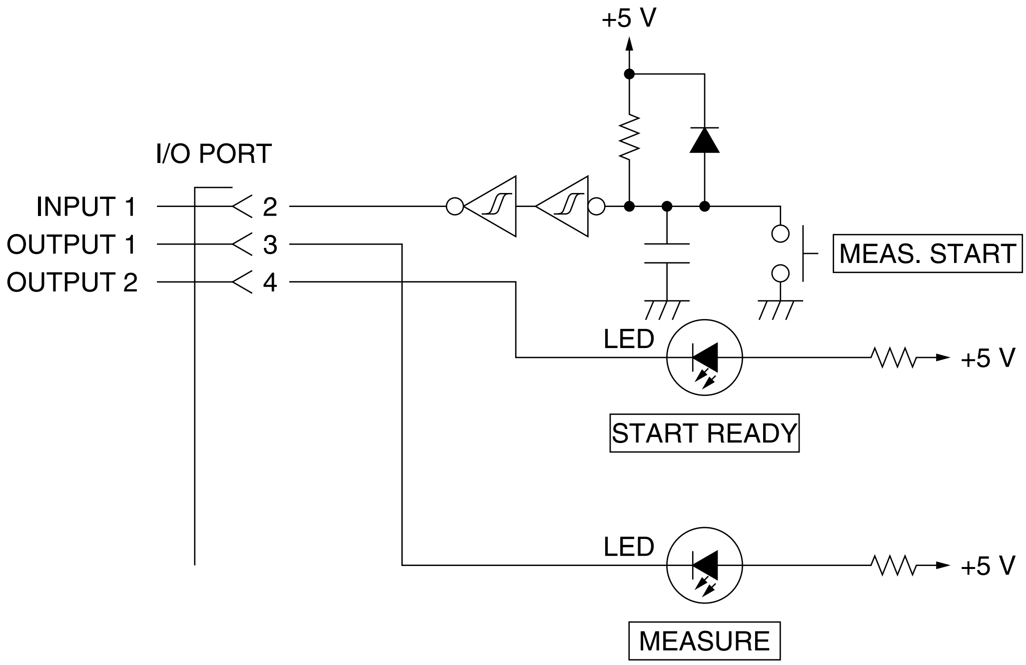
The measuring routine is repeatedly executed until an interrupt takes place. When an interrupt takes place, the processing moves to line 1,000. When the interrupt ends, the original measuring routine resumes.

9.5 I/O Port Use Example

(1) INPUT 1, OUTPUT 1 and OUTPUT 2

An example is shown below where INPUT 1 is used to enter the command to start measurements (MEAS, START), OUTPUT 1 is used to display the measurement and data processing and OUTPUT 2 is used to display the status of waiting for the measurement start.

<Circuit Example 1>



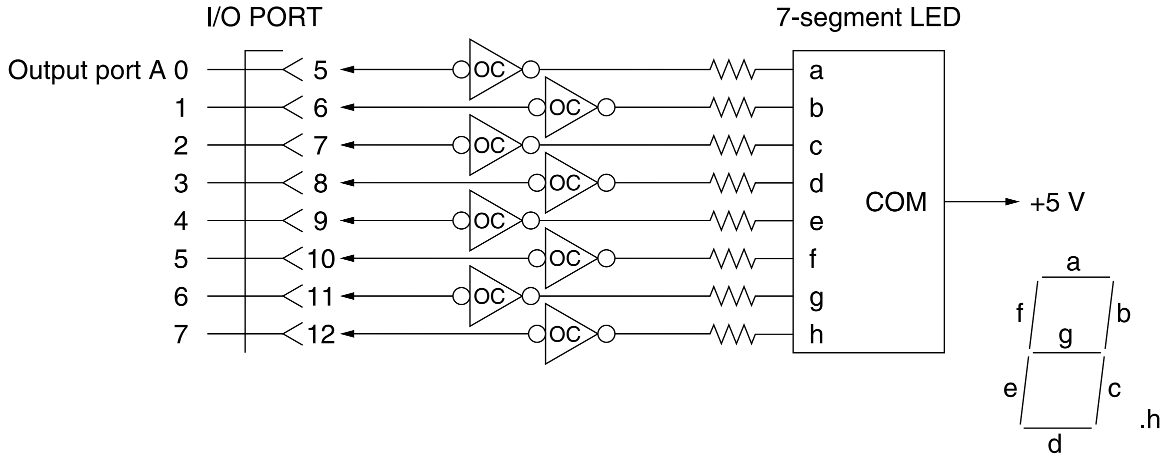
<Program Example 1>

```
1Ø EX0=Ø
:     ≀
:     Measurement preparation routine
:     ≀
1ØØ EXØ=2 .................................... Turning ON of the START READY lamp
11Ø C=EXØ .................................... Reading of the EXØ status
12Ø IF C=Ø GOTO 11Ø ............. Waiting until the MEAS.START switch is pressed
13Ø EXØ=1 .................................... Turning ON of the MEASURE lamp, OFF of the START READY lamp
:     ≀
:     Measurement preparation routine
:     ≀
2ØØ EXØ=Ø .................................... Turning OFF of the MEASURE lamp
21Ø STOP
```

(2) Output port A or output port B

An example is shown below where digits are output from the 7-segment LED for numeric display connected to output port A.

<Circuit Example 2>



<Program Example 2>

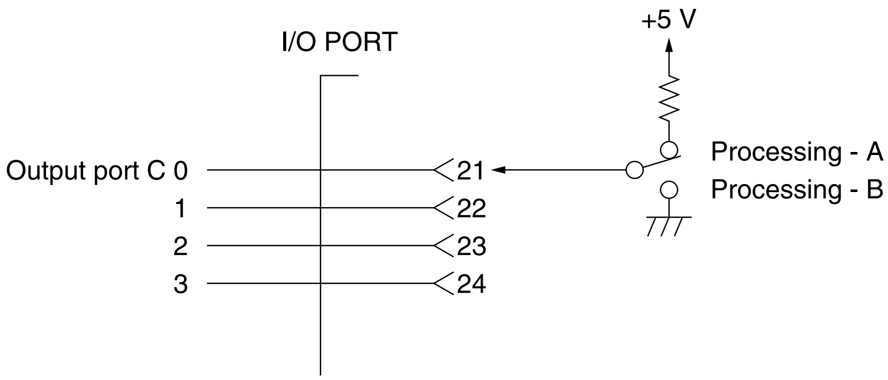
A program example is shown below where every time the MEAS. START switch in <Circuit Example 1> is pressed, digits 0 - 9, one digit each at a time, are displayed.

```
10Ø DIM D (1Ø)
2Ø  D (Ø) =$5C......................... Definition of the LED segment data
3Ø  D (1) =$06......................... Definition of the LED segment data
4Ø  D (2) =$5B......................... Definition of the LED segment data
5Ø  D (3) =$4F......................... Definition of the LED segment data
6Ø  D (4) =$66......................... Definition of the LED segment data
7Ø  D (5) =$6D......................... Definition of the LED segment data
8Ø  D (6) =$7D......................... Definition of the LED segment data
9Ø  D (7) =$Z7......................... Definition of the LED segment data
10Ø D (8) =$7F......................... Definition of the LED segment data
11Ø D (9) =$6F......................... Definition of the LED segment data
12Ø IOA=$Ø
13Ø N=Ø
14Ø EXØ=Ø
15Ø C=EXØ.............................. Reading of the EXØ status
16Ø IF C=Ø GOTO 15Ø................... Checking of the MEAS.START switch having been turned ON
17Ø IOA=D (N).......................... Display output
18Ø N=N+1
19Ø IF N=<9 GOTO 14Ø
20Ø STOP
```

(3) I/O port C or I/O port D

An example is shown below where switching bit 0 of I/O port C between 0 and 1 changes the processing routine.

<Circuit Example 3>



<Program Example 3>

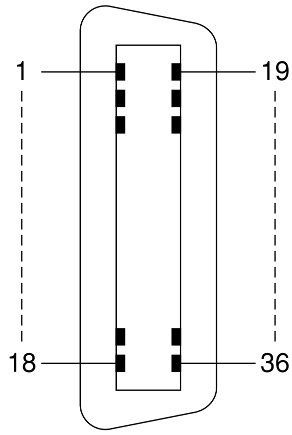
Setting a switch at processing A and B and pressing the MEAS. START switch in <Circuit 1> prompt checking of PORT C and branching of the processing.

```
10     EIO=0
20     EX0=0
30     C=EX0
40     IF C=0 GOTO 30
50     D=IOC
60     IF D=1 GOTO 200
:    ≀
:   Processing-A
:    ≀
100    STOP
200    REM
:    ≀
:   Processing-B
:    ≀
250    STOP
```

9.6 Item Name and Applicable Connector of I/O Port Connector

The item name of the interface used for the I/O port is RC30-36R (manufactured by Hirose Electric). The connector which complies with the standards of this interface is RC30-36P (manufactured by Hirose Electric).

Connector pin No. array



Anritsu

MS4630B

Network Analyzer

Vol. 2 Remote Operation

Operation Manual

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

Anritsu

MS4630B

NETWORK ANALYZER Vol. 2 REMOTE OPERATION OPERATION MANUAL

ANRITSU CORPORATION 5-10-27, Minamiazabu, Minato-ku, Tokyo 106-8570 Japan / Phone: 81-3-3446-1111

Document No.: M–W1535AE

Printed in Japan